Network Analysis with SiLK

Ron Bandes

SEI/CERT Network Situational Awareness

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
06 JAN 2015	**N/A**	

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Network Analysis with SiLK	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Bandes /Ron	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**141**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Copyright 2014 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0002038

Outline — 1

Introduction: SiLK

Network flow

Basic SiLK tools

Advanced SiLK tools

Summary

What is Network Flow?

- A log of all network activity
- Not a recording of all packets
- A record of metadata from related packets
  - Similar to a phone bill (call detail record)
- Content of messages is *not* recorded
  - Much, much more compact
    - Can retain longer
    - Less processing
  - Increased privacy

What SiLK Does

Retrospective analysis

- most useful for analysing past network events
- may feed an automated report generator
- good for forensics (what happened ***before*** the incident?)

Descriptive analysis – profiling/categorizing

Exploratory analysis – looking for the unusual

Optimized for extremely large data collections

- Very compact record format
- Large amount of history can stay online.
- Can be processed much more quickly than packets

Modes of Inquiry

Detect	Discover
example: Snort®	example: SiLK™
Operate like Tech Support	Operate like Quality Assurance
Response	Exploring
Alert	Question
Relax between detections	Continuous security improvement Produce new indicators Shorten detect/response time

Got a Question? Flow Can Help

What's on my network?

What happened before the event?

Where are policy violations occurring?

What are the most popular web servers?

By how much would volume be reduced with a blacklist?

Do my users browse to known infected web servers?

Do I have a spammer on my network?

When did my web server stop responding to queries?

Who uses my public servers?

Outline — 2

Introduction: SiLK

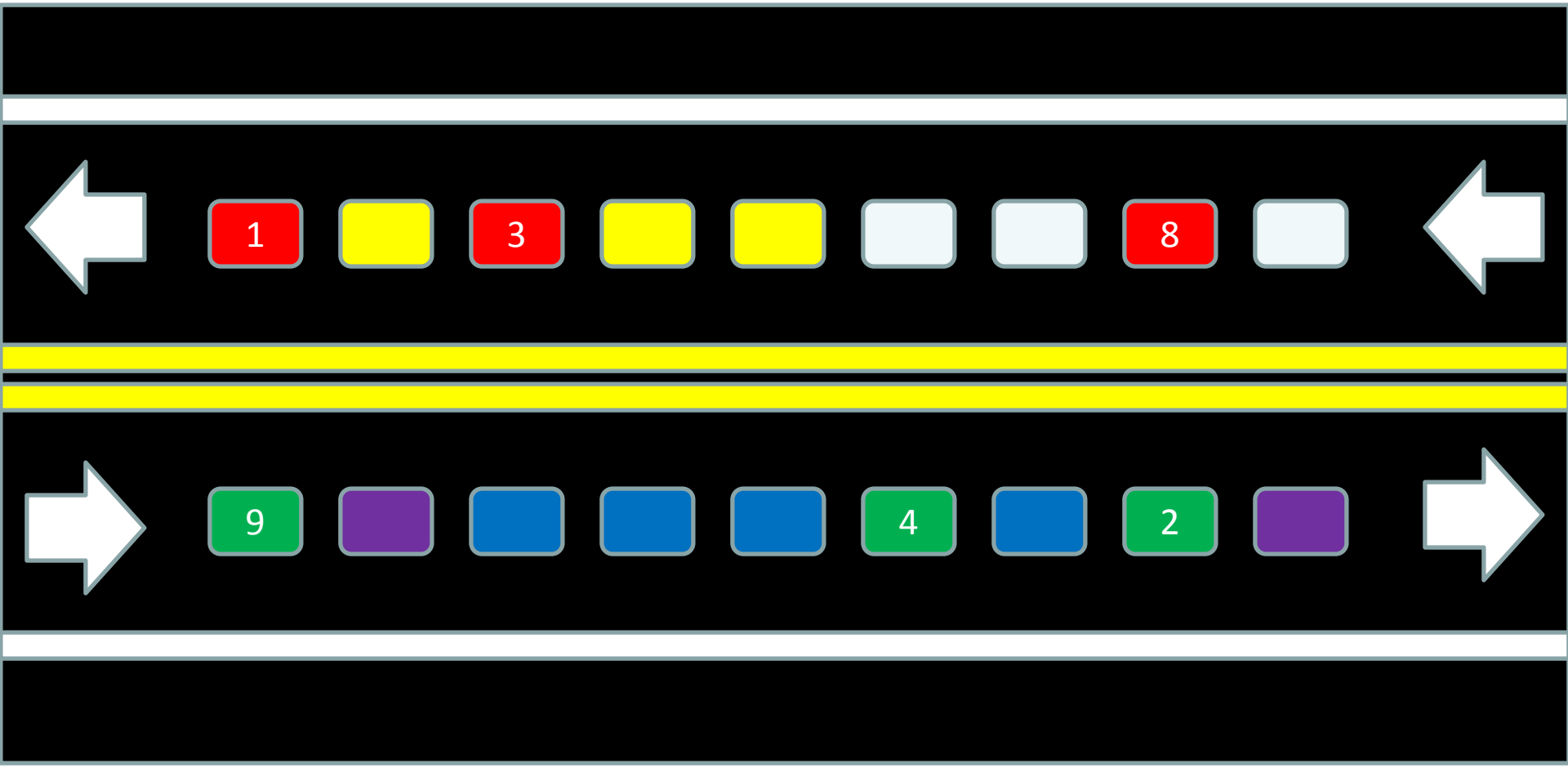
Network flow

Basic SiLK tools

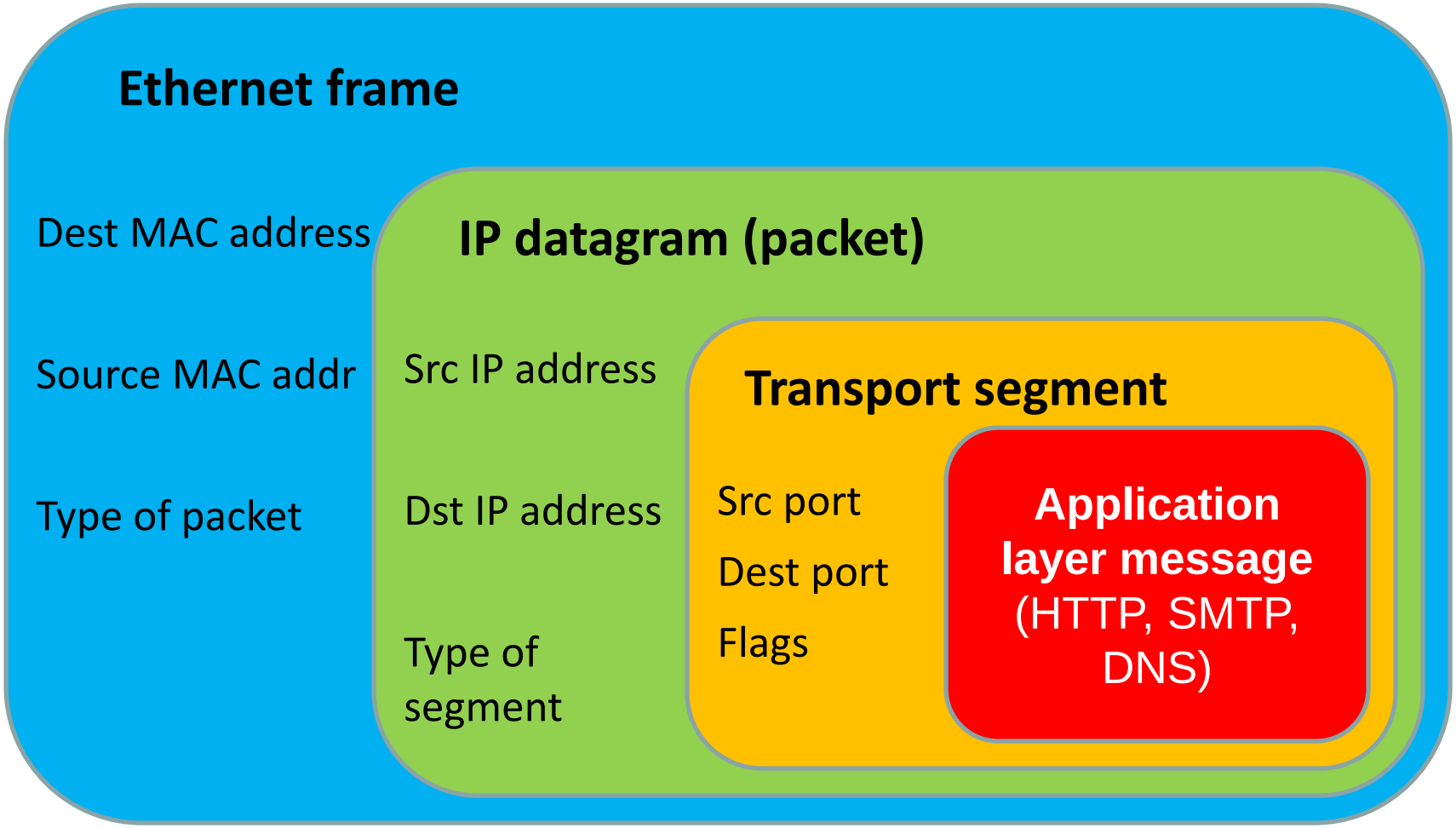
Advanced SiLK tools

Summary

Unidirectional Flows (Uniflows)



Packet Encapsulation



Two TCP/IP Sockets Make a Connection

TCP/IP SOCKET
IP address: 10.0.0.1
L4 protocol: TCP
High-numbered ephemeral port #

TCP/IP SOCKET
IP address: 203.0.113.1
L4 protocol: TCP
Low-numbered Well-Known-Port #

Client Connection Server

Network Flow versus NetFlow

Network Flow—a generic term for the summarization of packets related to the same flow or connection into a single record

NetFlow™—A Cisco trademarked set of format specifications for storing network flow information in a digital record

IPFIX—a format specification from the IETF for flow records, an extension of Cisco NetFlow v9

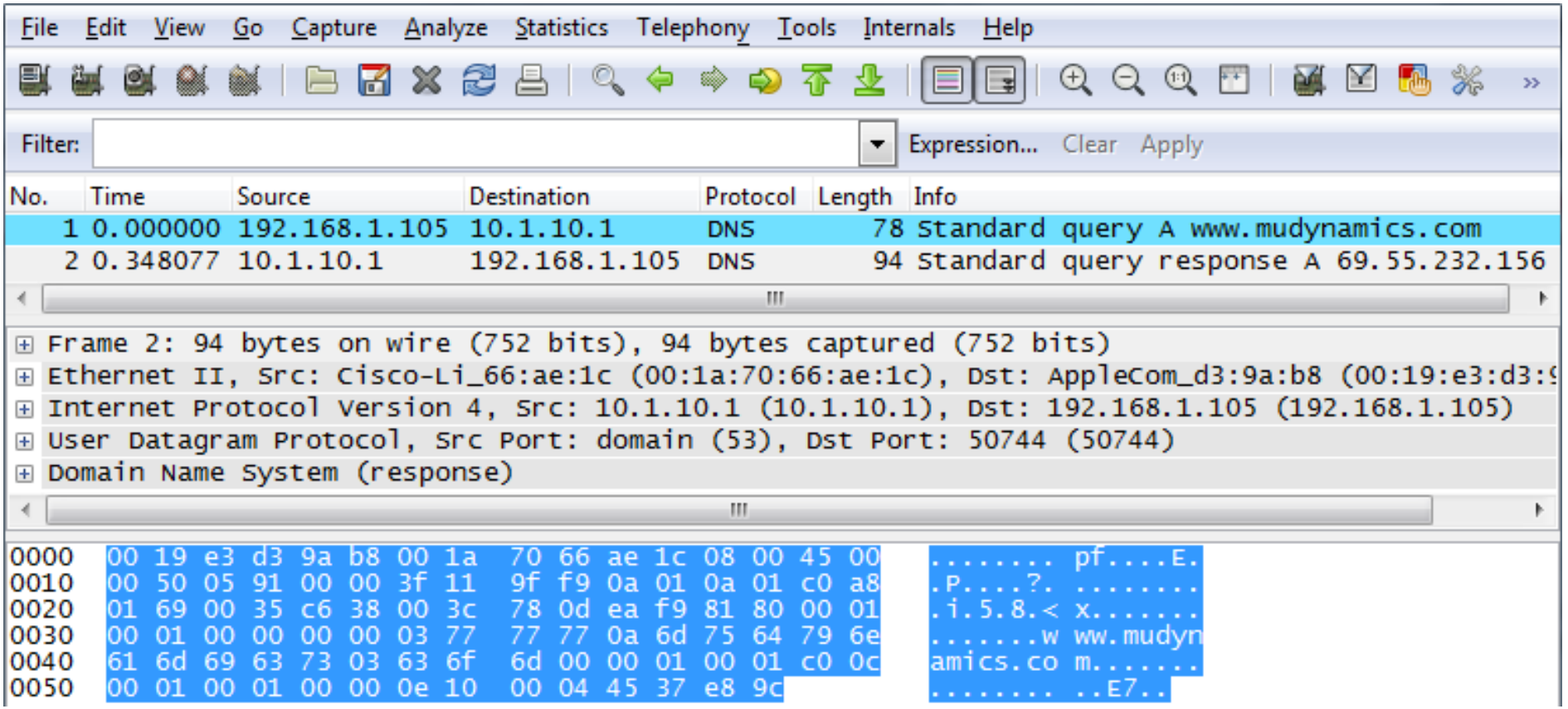
SiLK—Another set of format specifications for flow records and other related data, plus the tool suite to process that data

What's in a Record?

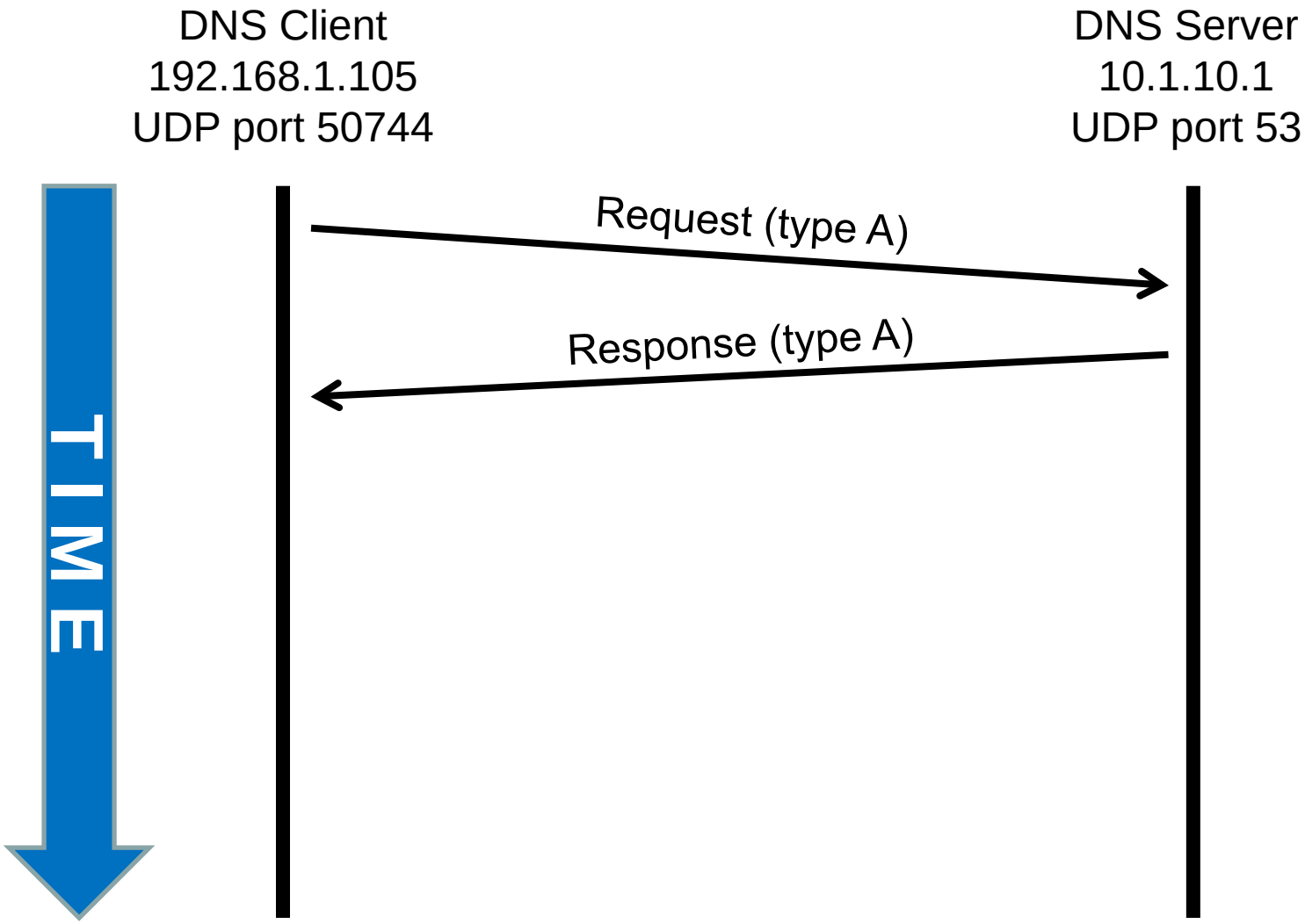
Fields found to be useful in analysis:

- source address, destination address
- source port, destination port (Internet Control Message Protocol [ICMP] type/code)
- IP [transport] protocol
- bytes, packets in flow
- accumulated TCP flags (all packets, first packet)
- start time, duration (milliseconds)
- end time (derived)
- sensor identity
- flow termination conditions
- application-layer protocol

DNS packets viewed in Wireshark



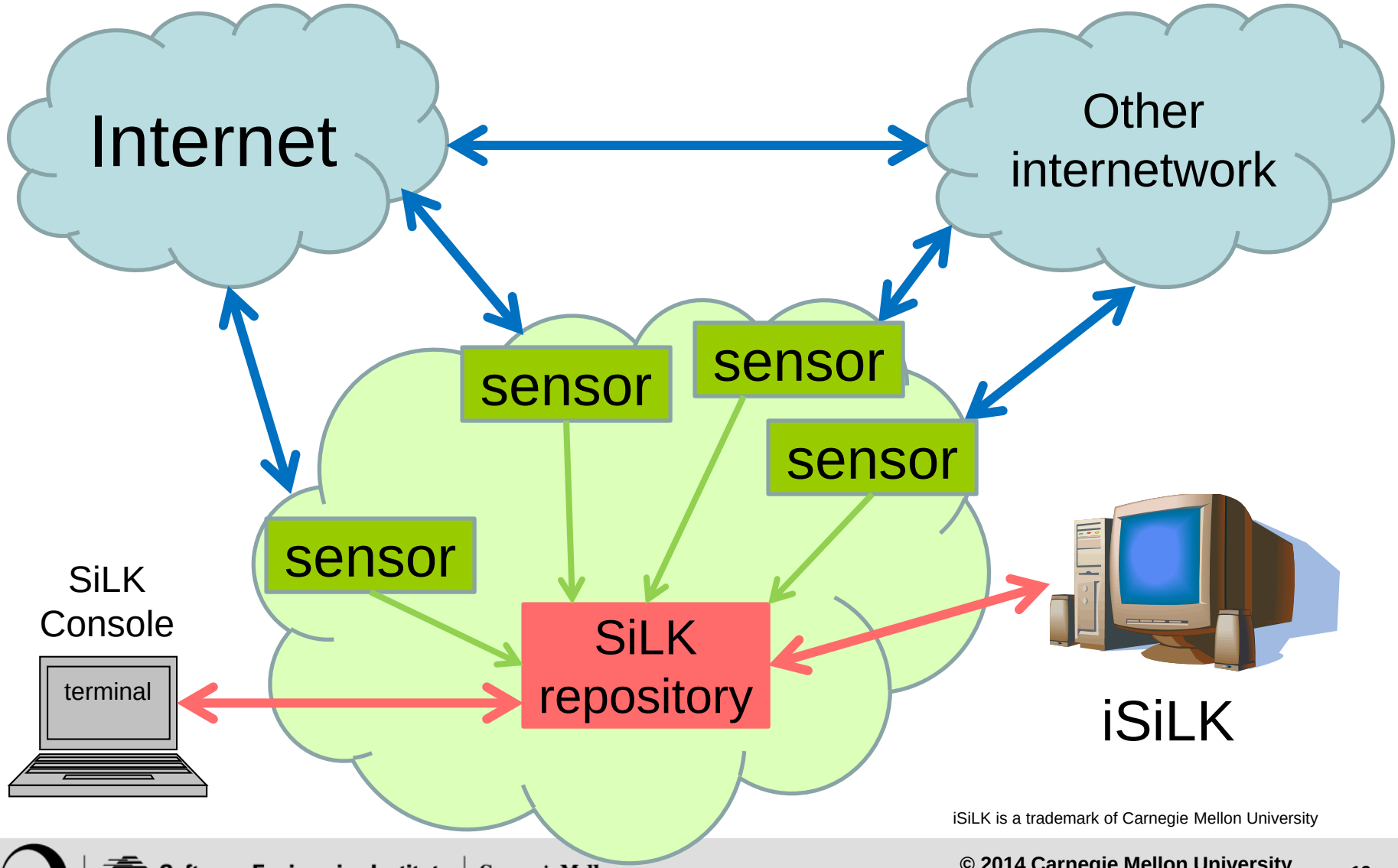
Sequence Diagram



SiLK tool (rwcut) output

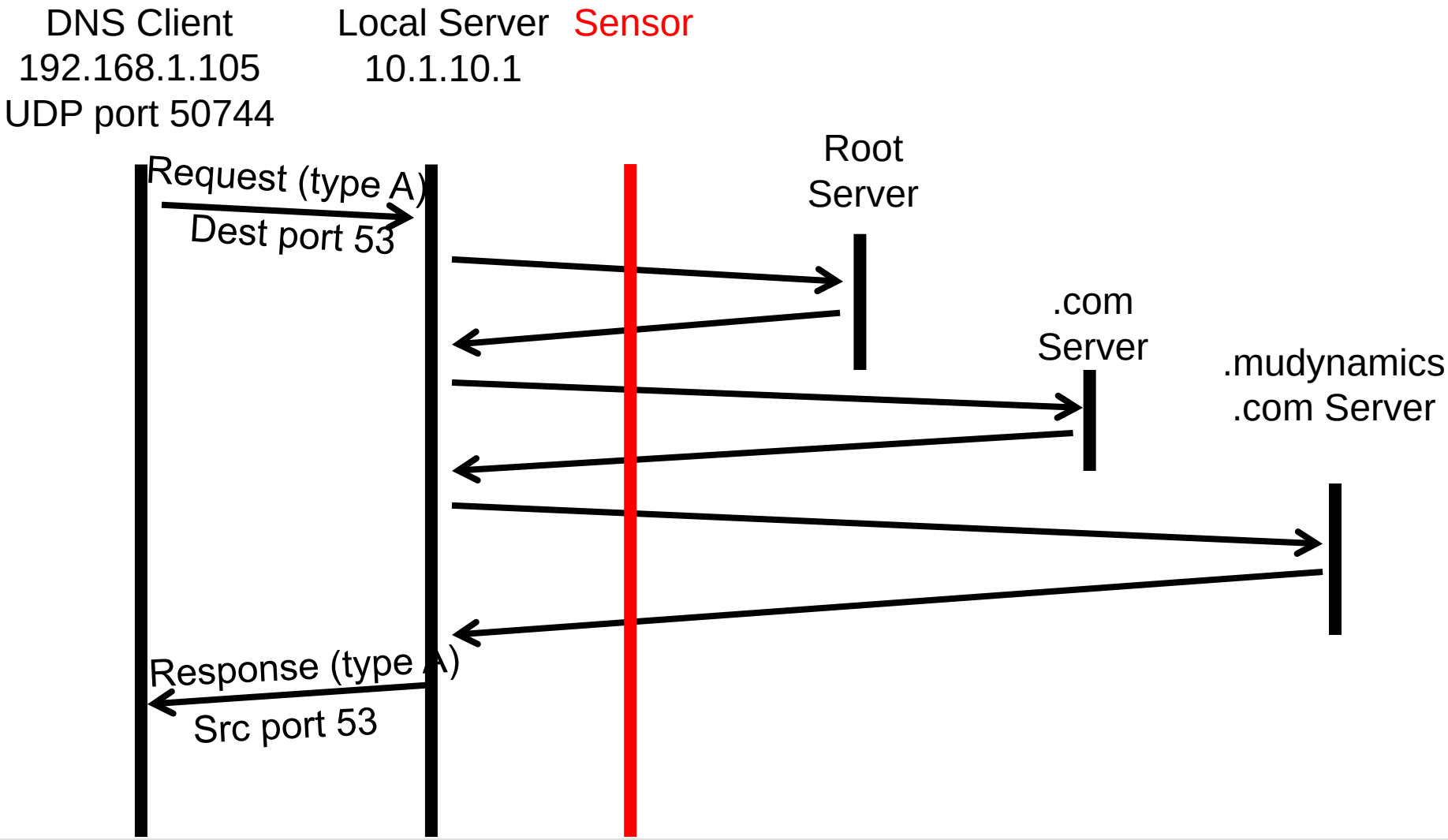
```
          sIP|          dIP|sPort|dPort|pro|packets|bytes|sensor|type|
192.168.1.105|    10.1.10.1|50744|   53| 17|      1|   64|    S1| out|
    10.1.10.1|192.168.1.105|   53|50744| 17|      1|   80|    S1|  in|
```

Network Monitoring

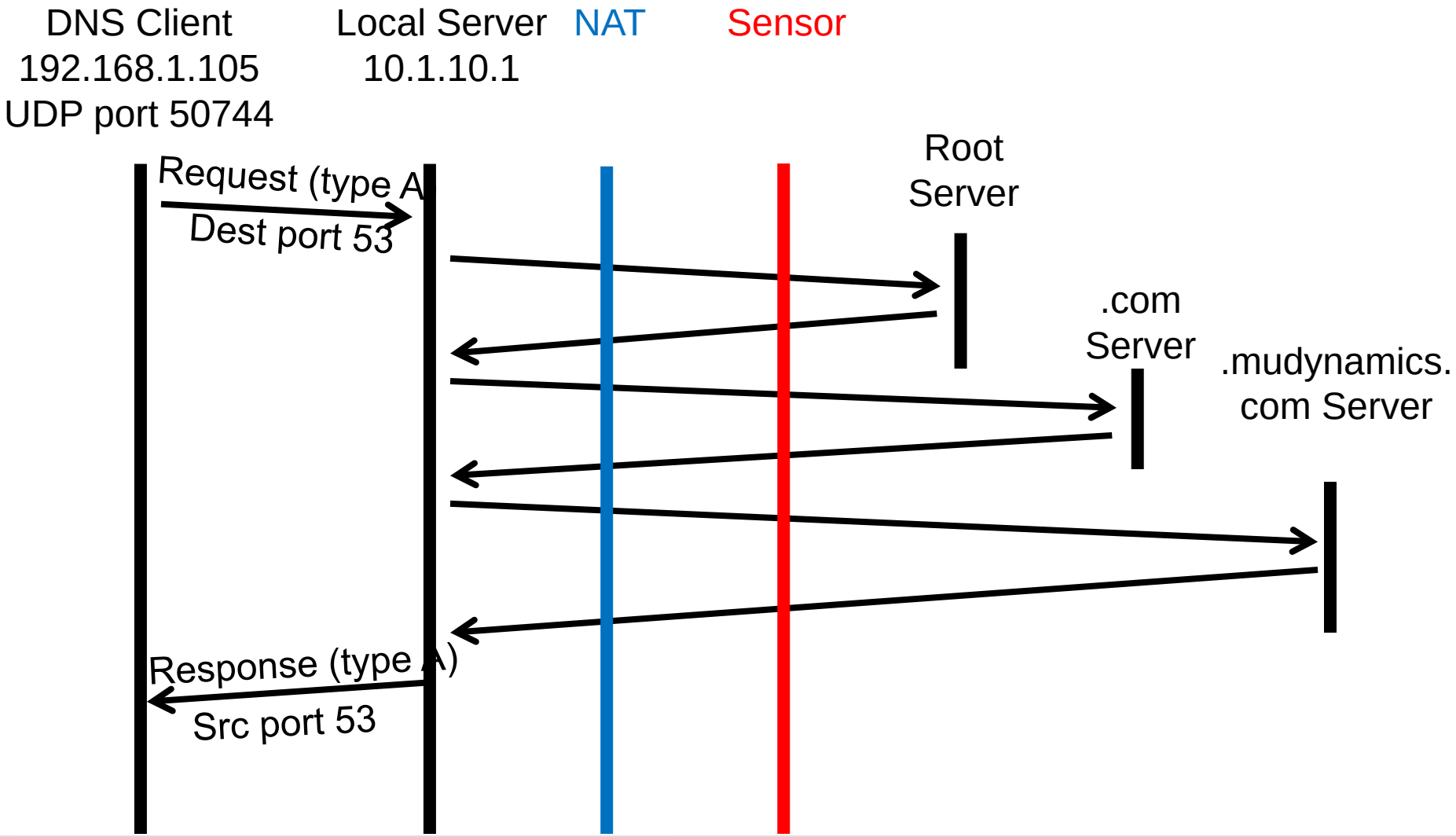


iSiLK is a trademark of Carnegie Mellon University

Realistic Sequence Diagram



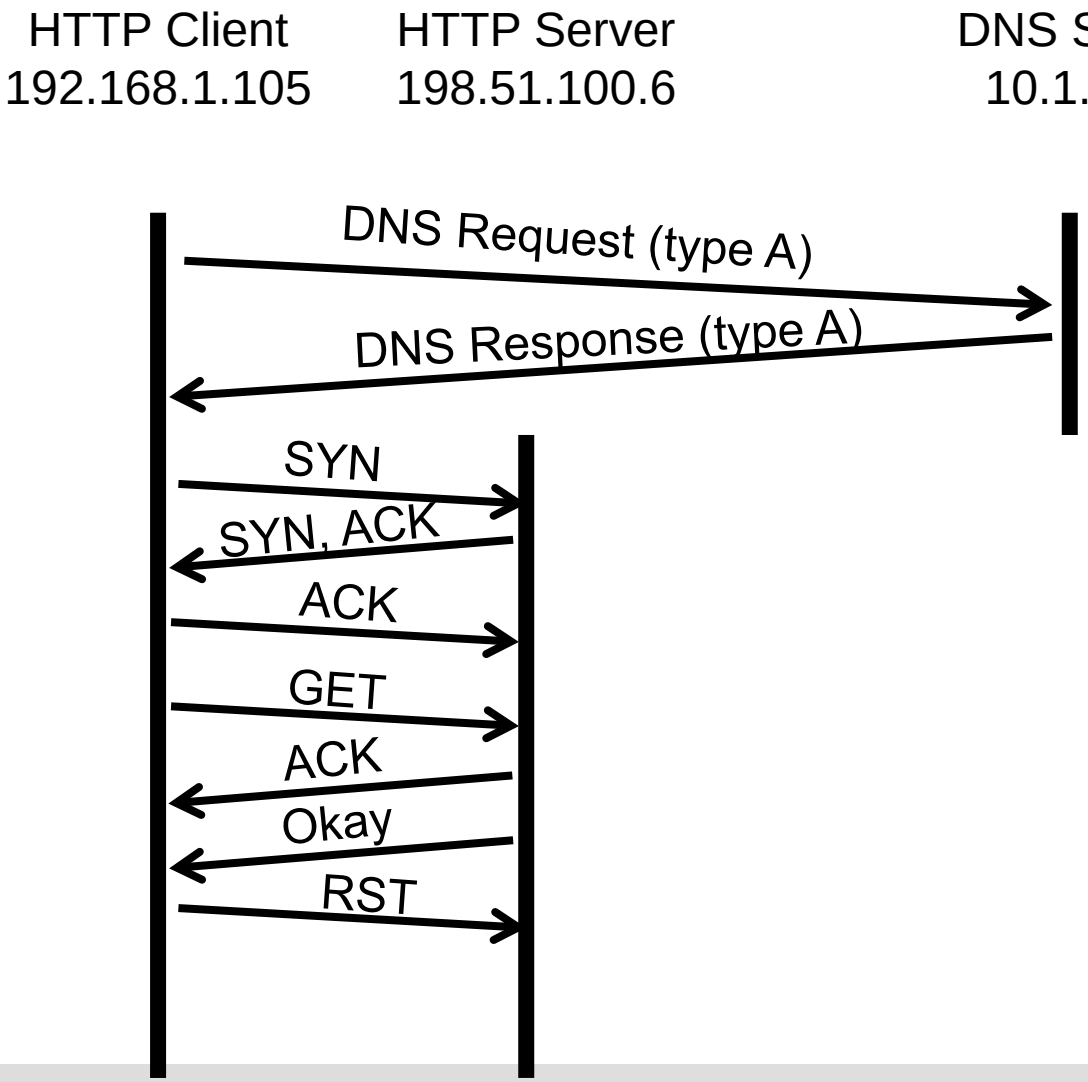
More Realistic Sequence Diagram



What is this? — 1

```
          sIP|          dIP|sPort|dPort|pro|packets|flags|initF|  type|
192.168.1.105|    10.1.10.1|50744|   53| 17|      1|     |     |   out|
    10.1.10.1|192.168.1.105|   53|50744| 17|      1|     |     |    in|
192.168.1.105| 198.51.100.6|49152|   80|  6|      4| SRPA| S   |outweb|
 198.51.100.6|192.168.1.105|   80|49152|  6|      3| S PA| S  A| inweb|
```

HTTP Sequence Diagram



What Is This? — 2

sIP	dIP	sPort	dPort	pro	packets	bytes	flags
30.22.105.250	71.55.40.253	52415	25	6	22	14045	FSRPA
71.55.40.253	30.22.105.250	25	52415	6	19	1283	FS PA
30.22.105.250	71.55.40.253	52415	25	6	1	40	R

What Is This? — 3

sIP	dIP	pro	packets	bytes	sTime
99.217.139.155	177.252.24.89	1	2	122	2010/12/08T00:04:30.172
99.217.139.155	177.252.149.249	1	2	122	2010/12/08T00:04:37.302
99.217.139.155	177.252.24.52	1	2	122	2010/12/08T00:04:37.312
99.217.139.155	177.252.24.127	1	2	122	2010/12/08T00:04:58.363
99.217.139.155	177.252.24.196	1	2	122	2010/12/08T00:05:04.327
99.217.139.155	177.252.149.30	1	2	122	2010/12/08T00:05:09.242
99.217.139.155	177.252.149.173	1	2	122	2010/12/08T00:05:12.174
99.217.139.155	177.252.24.13	1	2	122	2010/12/08T00:05:14.114
99.217.139.155	177.252.24.56	1	2	122	2010/12/08T00:05:15.383
99.217.139.155	177.252.24.114	1	2	122	2010/12/08T00:05:18.228
99.217.139.155	177.252.202.92	1	2	122	2010/12/08T00:05:22.466
99.217.139.155	177.252.202.68	1	2	122	2010/12/08T00:05:23.497
99.217.139.155	177.252.24.161	1	2	122	2010/12/08T00:05:30.256
99.217.139.155	177.252.202.238	1	2	122	2010/12/08T00:05:33.088

What Is This? — 4

sIP	dIP	sPort	dPort	pkts	bytes	flags	sTime
88.187.13.78	71.55.40.204	40936	80	83	3512	FS PA	2010/12/08T11:00:01
71.55.40.204	88.187.13.78	80	40936	84	104630	FS PA	2010/12/08T11:00:01
88.187.13.78	71.55.40.204	40938	80	120	4973	FS PA	2010/12/08T11:00:04
71.55.40.204	88.187.13.78	80	40938	123	155795	FS PA	2010/12/08T11:00:05
88.187.13.78	71.55.40.204	56172	80	84	3553	FS PA	2010/12/08T12:00:02
71.55.40.204	88.187.13.78	80	56172	83	103309	FS PA	2010/12/08T12:00:02
88.187.13.78	71.55.40.204	56177	80	123	5093	FS PA	2010/12/08T12:00:05
71.55.40.204	88.187.13.78	80	56177	124	157116	FS PA	2010/12/08T12:00:05

It's All a Matter of Timing

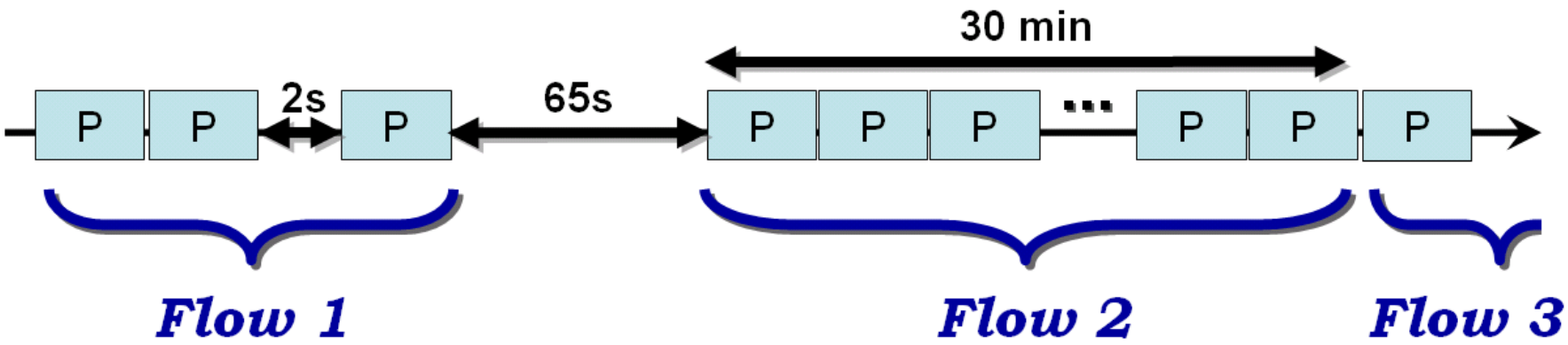
The flow buffer needs to be kept manageable.

Idle timeout

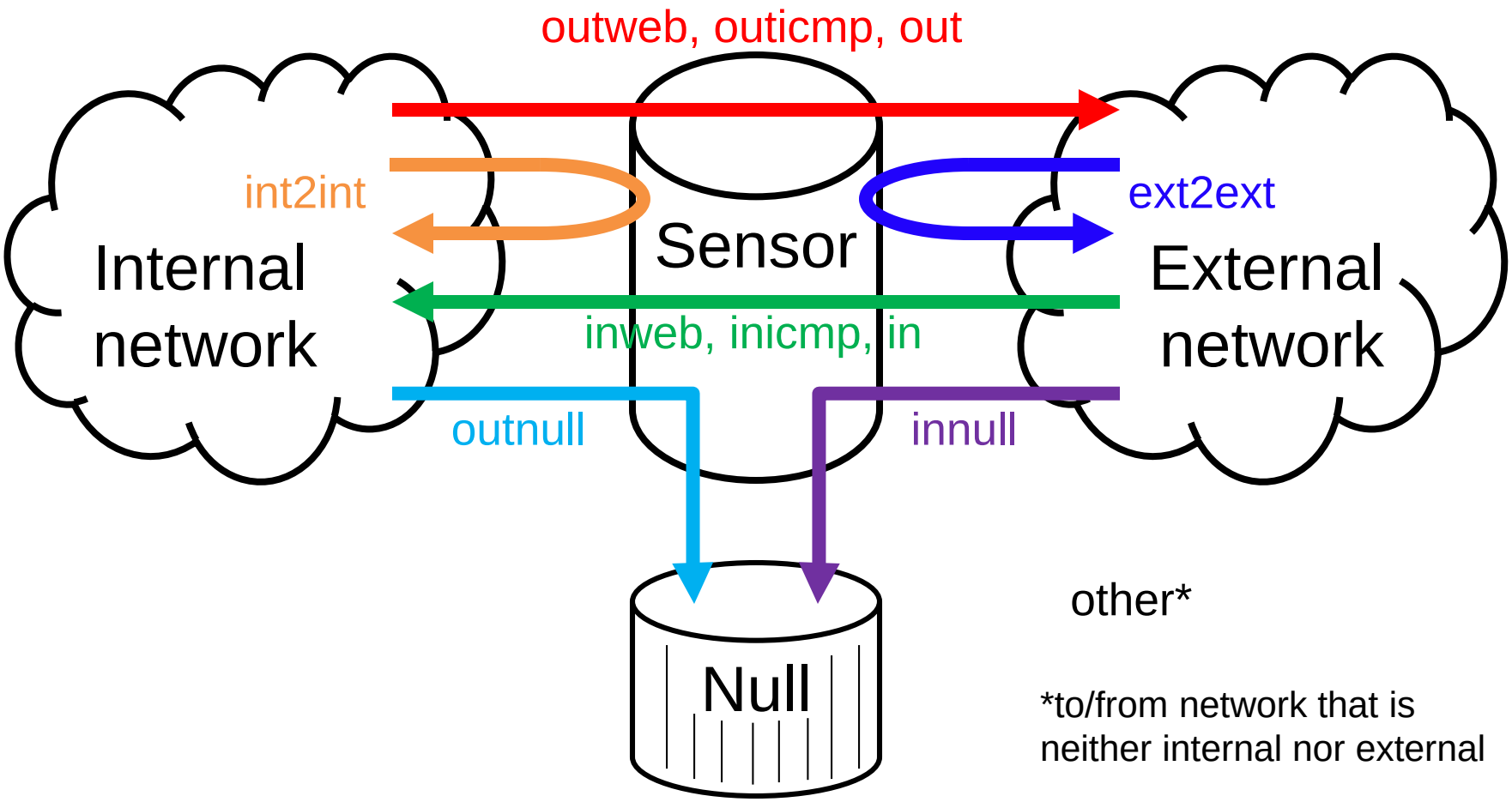
- If there is no activity within 30 seconds (configurable), flush the flow.

Active timeout

- Flush all flows open for 30 minutes (configurable).



SiLK Types



SiLK Types in SiLK

Type	Description
inweb, outweb	Inbound/outbound TCP ports 80, 443, 8080
innull, outnull	Inbound/outbound filtered traffic
inicmp, outicmp	Inbound/outbound IP protocol 1
in, out	Inbound/outbound not in above categories
int2int, ext2ext	Internal to internal, external to external
other	Source not internal or external, or destination not internal, external, or null

Names in **bold** are default types

Outline — 3

Introduction: SiLK

Network flow

Basic SiLK tools

Advanced SiLK tools

Summary

UNIX / Linux commands

System prompt

Info + prompt character

e.g., **~ 101>**

User command

command name **rwfilter** (case sensitive)

options **-h --help -k2 --key=2**

arguments **results.rw**

redirections **> >> <**

pipe **|**

For example:

```
rwcut --all-fields results.rw >results.txt
rwcut --fields=1-6 results.rw | more
```

Some standard Linux commands

`ls` – list name & attributes of files and directories

`cd` – change the current working directory

`cat` – output the contents of a file

`more` and **`less`** – display a file one page at a time

`cut` – output only selected fields of a file

`sort` – reorder the records (lines) of a file

`wc` – word count (optionally, line count) of a file

`exit` – logout & terminate a terminal window

Linux Standard symbolic files

Standard In (**stdin**) – where normal (especially interactive) input comes from

Standard Out (**stdout**) – where normal/expected (especially interactive) output goes to

Standard Error (**stderr**) – where messages (especially unexpected) go to

Defaults:

stdin – keyboard

stdout – screen/window

stderr – screen/window

Defaults are overridden by redirections and pipes

Shell Scripts

Put a complicated command, pipeline, or sequence of pipelines into a script file.

- It saves your commands for reuse or learning
- It eases making changes

Use the GUI editor **`gedit`**, or the simple character editors **`joe`** and **`nano`** when on a SSH connection. Use **`vi`** (**`vim`**) to earn your geek badge. **`Vi`** or **`vim`** can be found on every Linux/UNIX system.

Name your shell script something like **`dothis.sh`**

Execute (run) your script: **`./dothis.sh`**

Collection, Packing, and Analysis

Collection of flow data

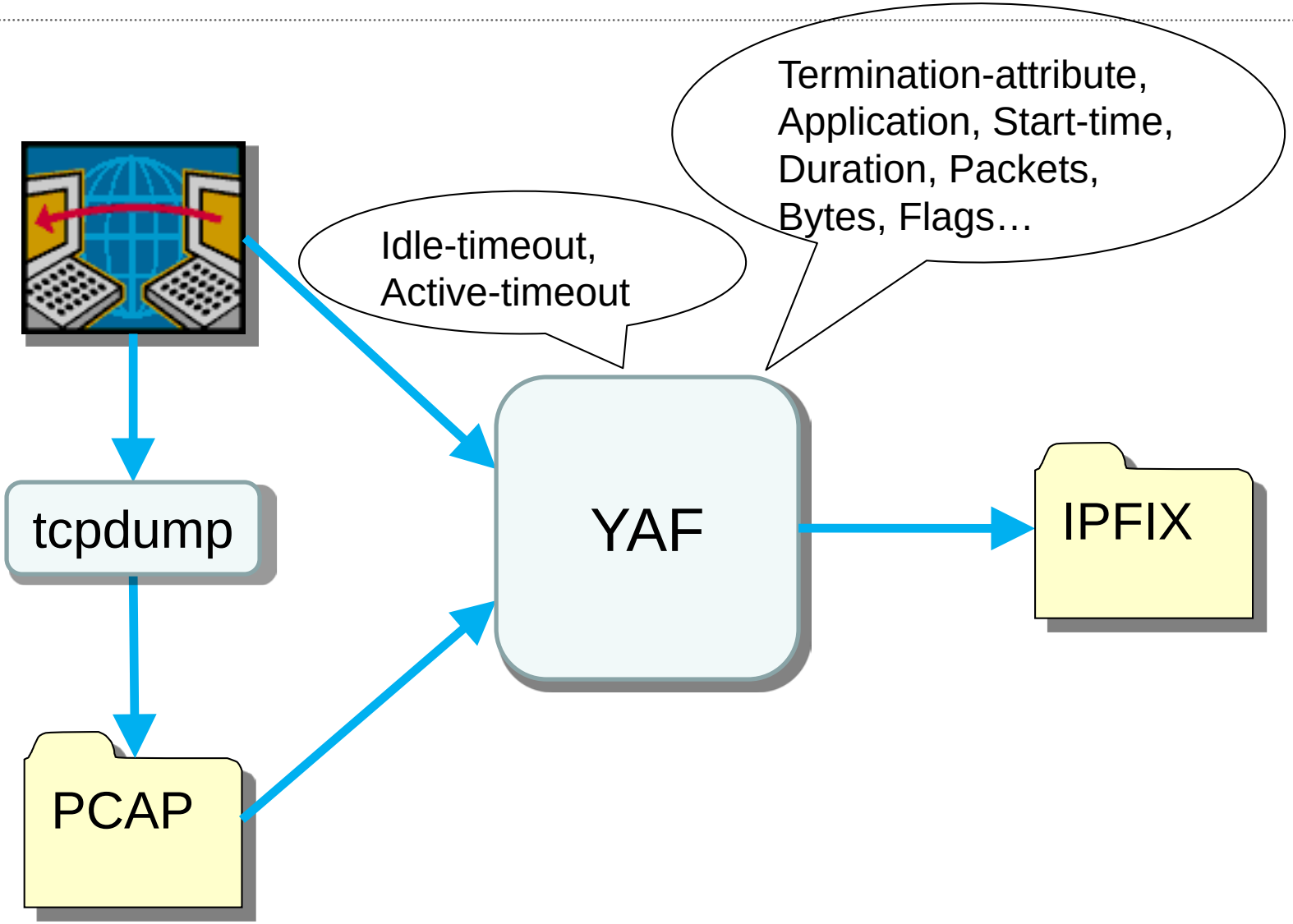
- Examines packets and summarizes into standard flow records
- Timeout and payload-size values are established during collection

Packing stores flow records in a scheme optimized for space and ease of analysis

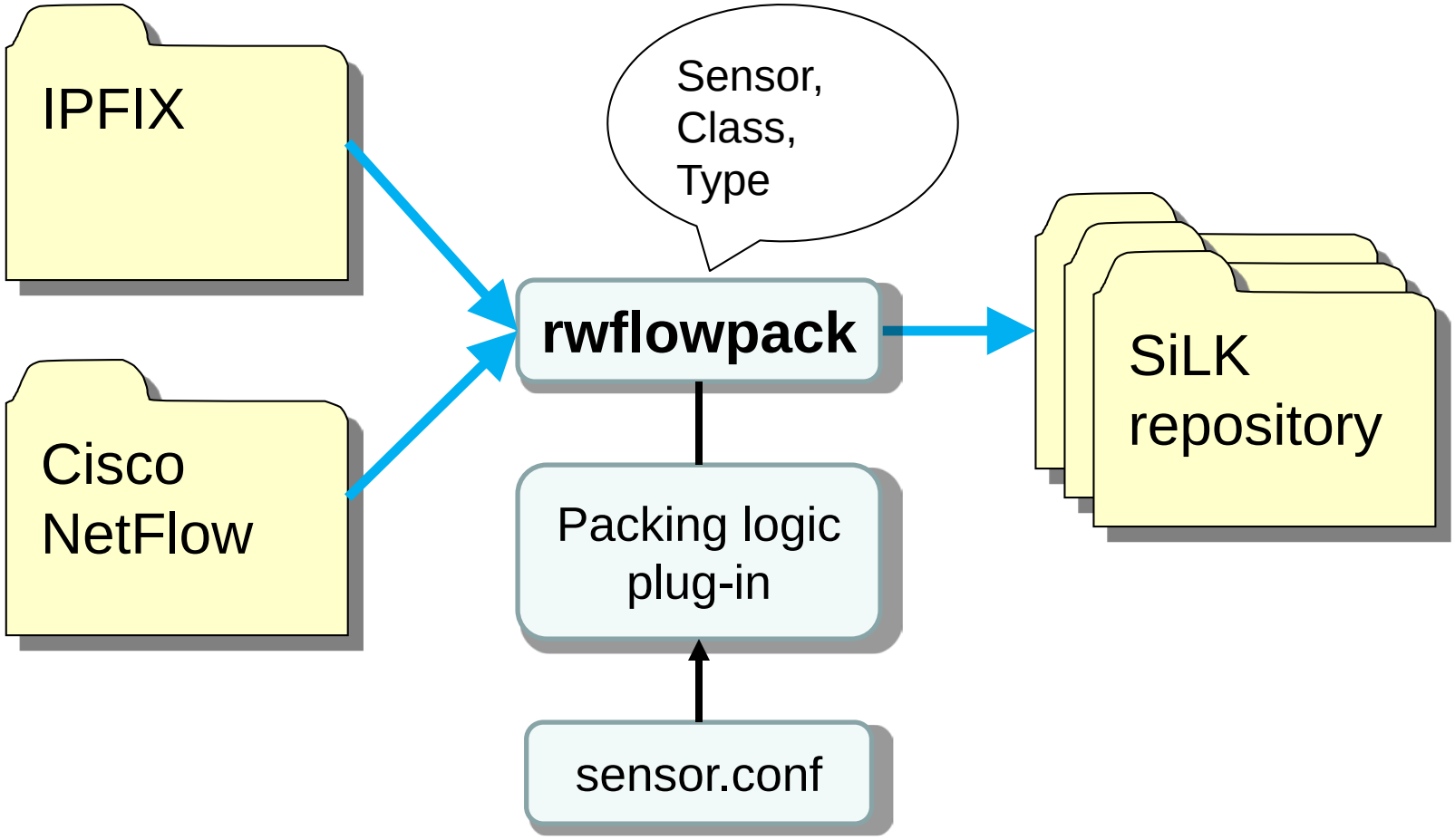
Analysis of flow data

- Investigation of flow records using SiLK tools

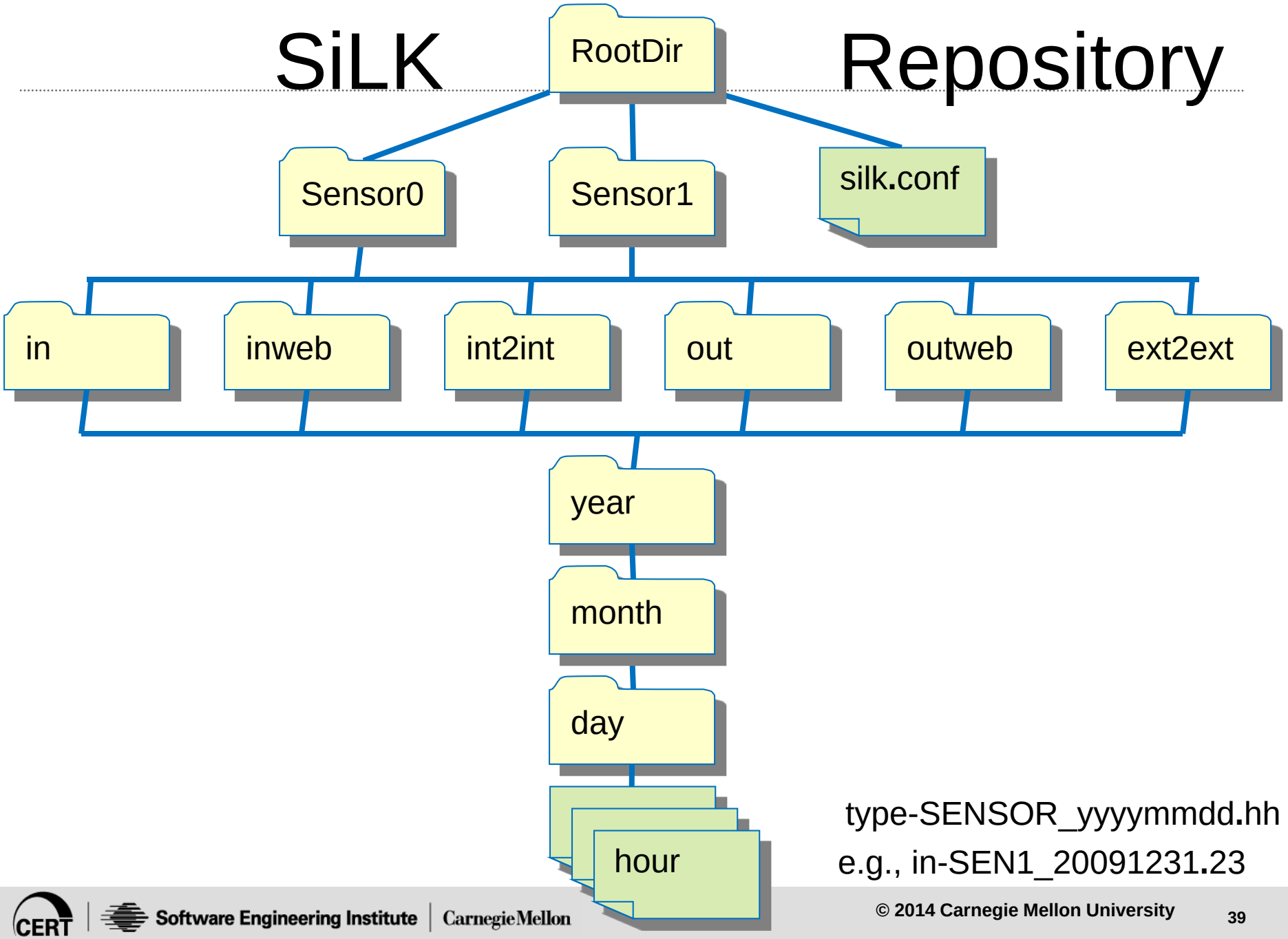
Collection



Packing



SiLK Repository



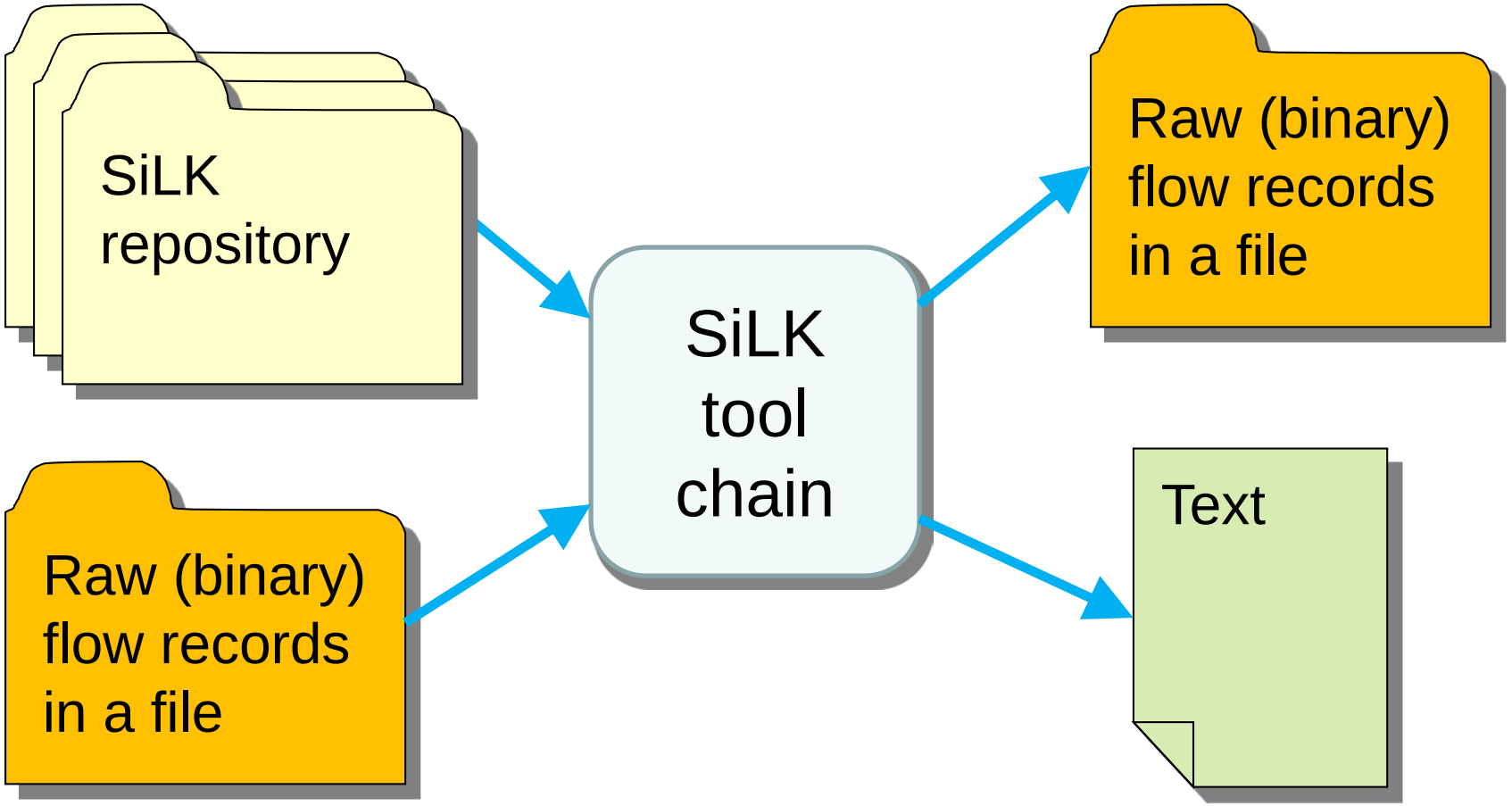
type-SENSOR_yyyymmdd.hh

e.g., in-SEN1_20091231.23

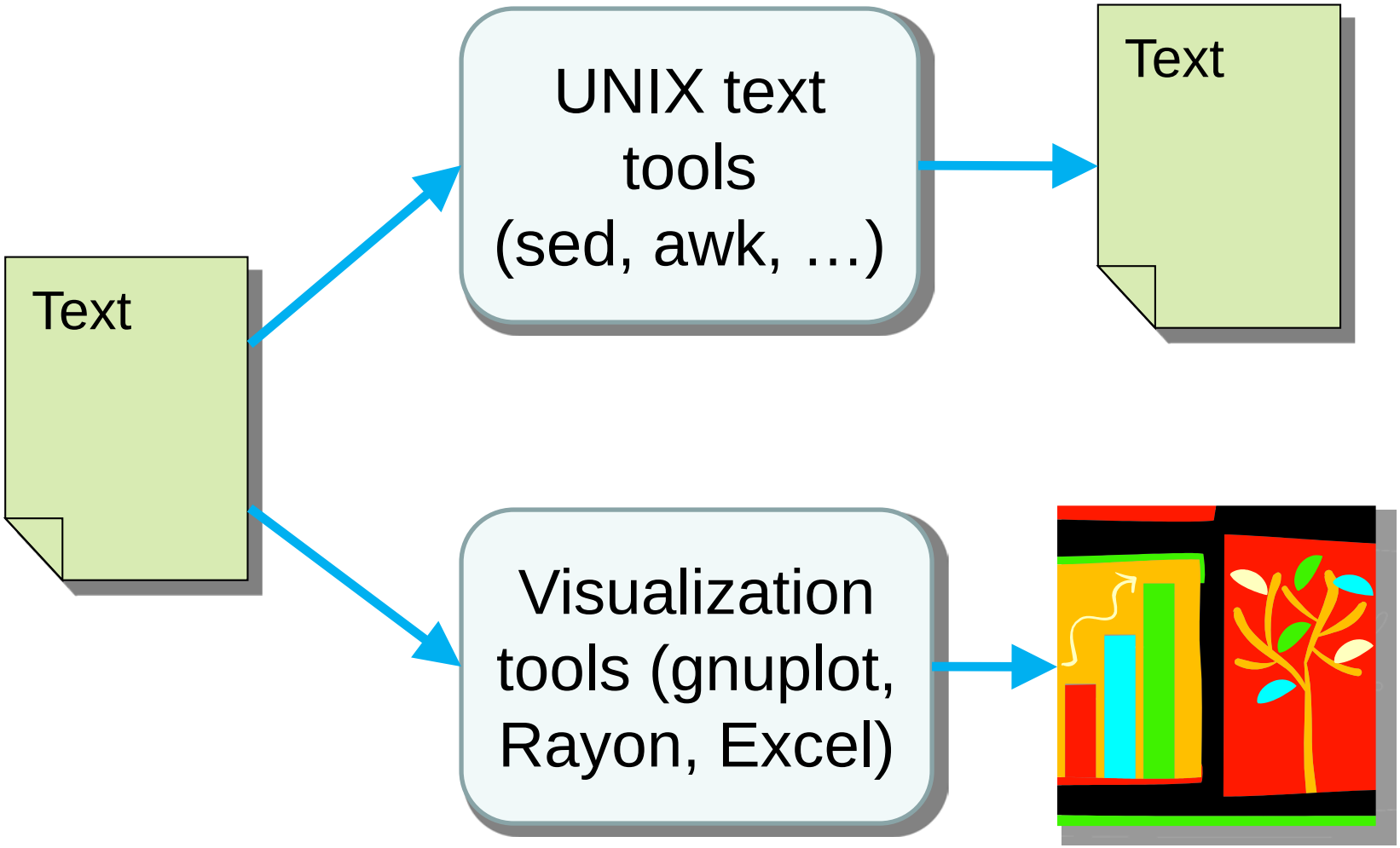
Linux Exercise

```
PS1='\W \!> '
export SILK_IPV6_POLICY=asv4
cd /data/bluered
ls -l silk.conf
less silk.conf   # type “q” to exit from less
cd
```

Analysis



Reporting



Rayon is a trademark of Carnegie Mellon University
Excel is a registered trademark of Microsoft Corporation

So Much to Do, So Little Time...

We can’t discuss all parameters for every tool.

Resources

- Analyst’s Handbook
- SiLK Reference Guide (hard-copy man pages)
- **`--help`** option
- **`man`** command
- **`http://tools.netsa.cert.org`**

What sensors are defined?

```
rwsiteinfo --fields=id-sensor,sensor
rwsiteinfo --fields=id-sensor,sensor,\
     describe-sensor
```

Basic SiLK Tools: `rwfileinfo`

`rwfileinfo` displays a variety of characteristics for each file format produced by the SiLK tool suite.

It is very helpful in tracing how a file was created and where it was generated.

rwfileinfo Example

```
[liveuser@livecd ~]$ rwfilter --sensor=S0 --type=in,out \
  --start=2009/4/21T15 --protocol=1 \
  --pass=icmprecords.rw
[liveuser@livecd ~]$ rwfileinfo icmprecords.rw
icmprecords.rw:
  format(id)          FT_RWIPV6ROUTING(0x0C)
  version             16
  byte-order          littleEndian
  compression(id)     lzo1x(2)
  header-length       176
  record-length       88
  record-version      1
  silk-version        3.9.0
  count-records       39
  file-size           963
  command-lines
    1  rwfilter --sensor=S0 --type=in,out
--start=2009/4/21T15 --protocol=1 --pass=icmprecords.rw
```

rwfileinfo --fields

All fields available to display

```
 1 format(id)
 2 version
 3 byte-order
 4 compression(id)
 5 header-length
 6 record-length
 7 count-records
 8 file-size
 9 command-lines
10 record-version
11 silk-version
12 packed-file-info
13 probe-name
14 annotations
15 prefix-map
16 ipset
17 bag
```

Basic SiLK Tools: `rwcut`

But I can't read binary...

`rwcut` provides a way to display binary records as human-readable ASCII:

- useful for printing flows to the screen
- useful for input to text-processing tools
- Usually you'll only need the **`--fields`** option.

sip	packets	type	flags
dip	bytes	in	initialflags
sport	sensor	out	sessionflags
dport	*scc*	dur	application
protocol	*dcc*	stime	attributes
class	nhip	*etime*	*itype & icode*

Field names in italics are *derived* fields

rwcut Default Display

By default

- sIP, sPort
- dIP, dPort
- protocol
- packets, bytes
- flags
- sTime, eTime, duration
- sensor

`--all-fields`

Pretty Printing SiLK Output

Default output is fixed-width, pipe-delimited data.

```
            sIP|            dIP|pro|pkts|bytes|
207.240.215.71| 128.3.48.203|  1|   1|   60|
207.240.215.71|  128.3.48.68|  1|   1|   60|
207.240.215.71|  128.3.48.71|  1|   1|   60|
```

Tools with text output have these formatting options:

- **`--no-titles`**: suppress the column headings
- **`--no-columns`**: suppress the spaces
- **`--column-separator:`** just change the bar to something else
- **`--delimited`**: combine above 3 options
- **`--legacy-timestamps`**: better for import to Excel

What do the data look like?

```
rwcut icmprecords.rw --fields=1-6
```

Try other values for **`--fields`**.

Try omitting the **`--fields`** option.

Why do we need rwcut?

```
cd
rwfilter --type=in \
   --start-d=2009/4/21T15 --proto=0- \
   --compress=none \
   --pass-dest=t20.rw --max-pass=20
ls -l t20.rw
rwfileinfo t20.rw
hexdump -C t20.rw  # any readable text?
rwcut --fields=1-6 t20.rw
```

Basic SiLK Tools: `rwsort`

Why sort flow records?

- Records are recorded as received, not necessarily in time order.
- Analysis often requires finding outliers.
- You can also sort on other fields such as IP address or port to easily find scanning patterns.
- It allows analysts to find behavior such as beaconing or the start of traffic flooding.

`rwsort` Options

`--fields` (same as rwcut) is required.

Input files are specified as positional arguments (default is stdin).

--output-path= specifies the output file (default is stdout.)

For improved sorts, specify a buffer size with --sort-buffer-size=.

For large sorts, specify a temporary directory with --temp-directory=. Temporary files stored in /tmp by default

```
rwsort t20.rw --fields=stime \
   --output-path=t20bystime.rw

rwsort t20.rw --fields=sip,sport,dport \
 | rwuniq --fields=sip,sport,dport --presorted \
          --value=dip-distinct
```

Basic SiLK Tools: `rwfilter`

Swiss Army knife logo is a registered trademark of Victorinox AG

rwfilter Syntax

General form

```
rwfilter {INPUT | SELECTION}
  PARTITION OUTPUT [OTHER]
```

Example call

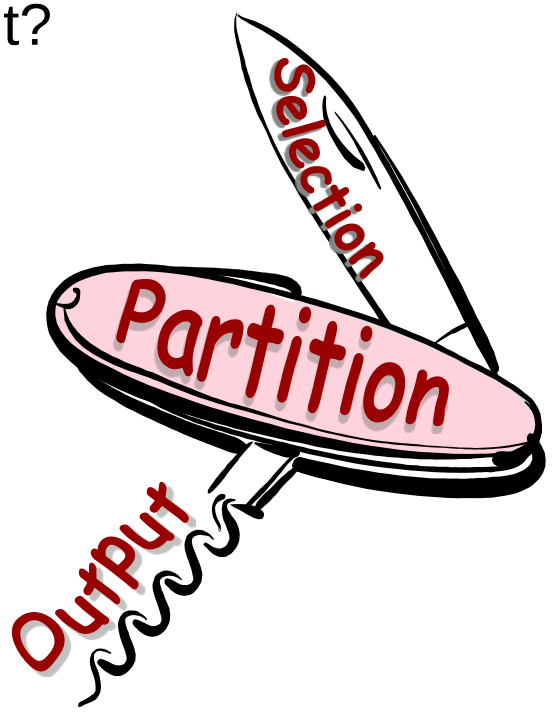
```
rwfilter --sensor=S0 --type=in \
  --start-date=2009/4/21T9 \
  --end-date=2009/4/21T16 \
  --protocol=0-255 --pass=workday-21.rw
```

`rwfilter` Command Structure

The rwfilter command requires three basic parts:

- **selection** criteria or **input** criteria (which files are input?)
  - repository: class, sensor, type, start/end date/hour
- **Partition** (which records pass my criteria? Which fail?)
  - filter options: Which flows do I really want?
- **output** options

Partitioning is the most complex part.



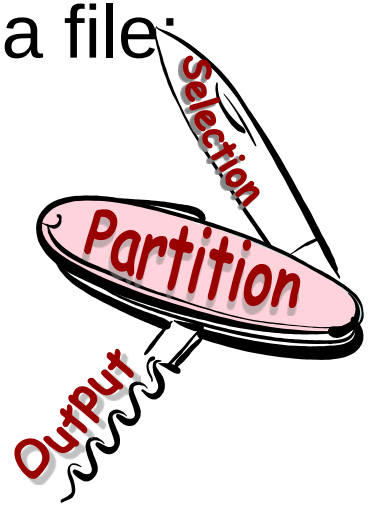
Selection and Input Criteria

Selection options control access to repository files:

- `--start-date=2009/4/21`
- `--end-date=2009/4/21T03`
- `--sensor=S0`
- `--class=all`
- `--type=in,inweb`

Alternatively, use input criteria for a pipe or a file:

- `myfile.rw`
- `stdin`
- useful for chaining filters through stdin/stdout



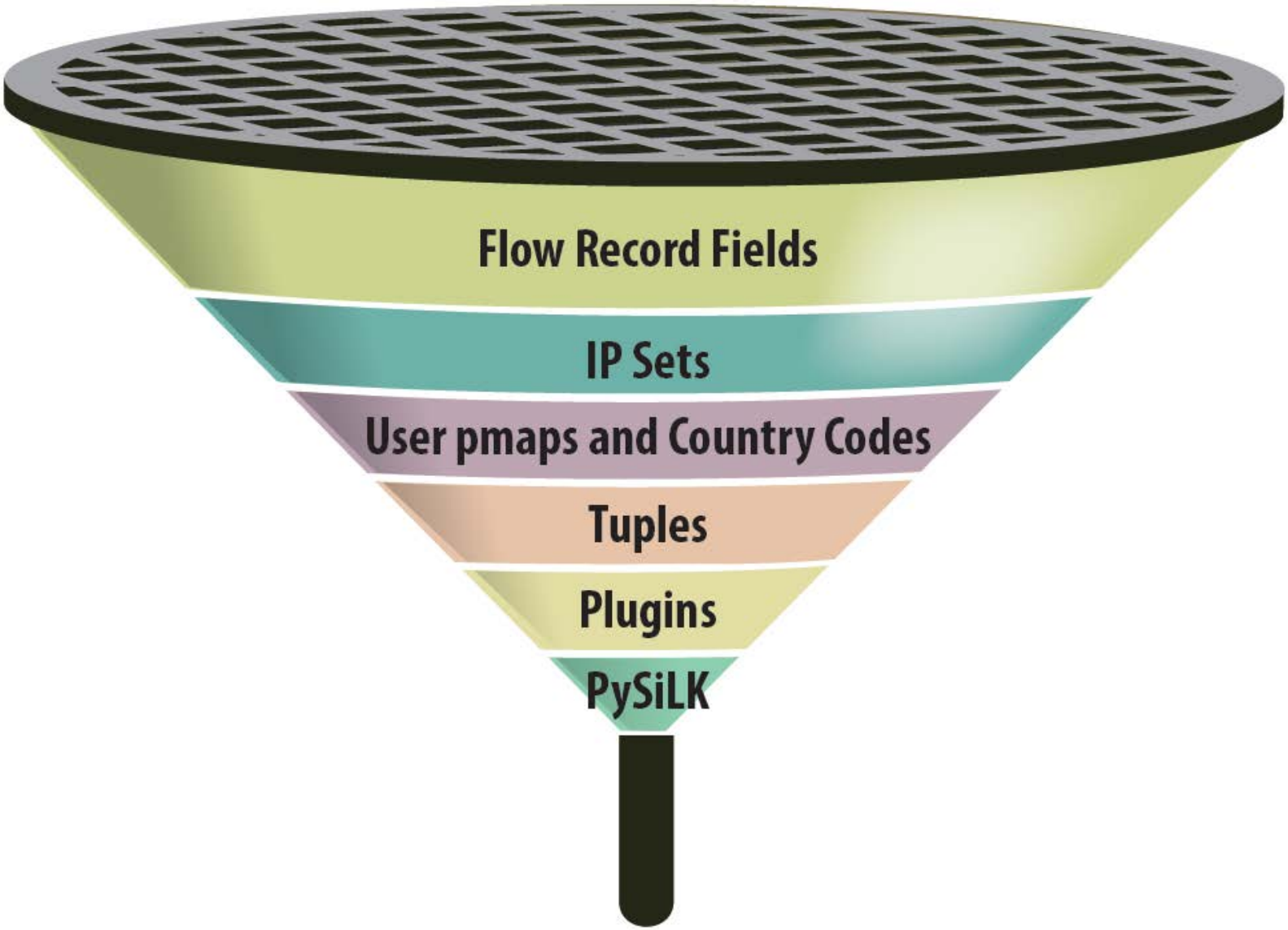
--start-date and --end-date

		--start-date		
		Hour	**Day**	**None**
--end-date	**Hour**	Hours in explicit range	Ignore end-date hour. Whole days.	Error
	Day	End-hour is the same as start-hour. #hours = 1, 25, 49, …	Whole days.	Error
	None	1 hour	1 day	Current day to present time.

How Many Files are Selected?

#Files = Sensors
x Types
x Hours
– missing files

`rwfilter` Partitioning Parameters



Basic Partitioning Options

- Simple numeric fields: ports, protocol, ICMP Type
- Specified IP addresses, CIDR blocks, & wildcards
- Sets of IP addresses
- Combinations of key fields – Tuples

Simple Numeric Key Fields

```
--protocol=
--sport= --dport= --aport=      # source, dest, any


--protocol=6,17                 # TCP or UDP
--protocol=1-5,7-16,18-         # not TCP or UDP
--protocol=0-                   # all protocols
--dport=80,443                  # HTTP or HTTPS
--sport=6000-6063,9100-9107     # X11 or JetDirect
--aport=20,21                   # FTP
--sport=0-1023                  # Well Known Ports
```

ICMP Types and Codes

--icmp-type major type of ICMP message

--icmp-code sub-type of ICMP message

--icmp-type=0,8 # ping request & reply

--icmp-type=3 --icmp-code=4 # fragm’n needed

Specified IP address, CIDR block, or wildcard

--**s**address= --**d**address= --any-address=

--not-**s**address= --not-**d**address= --not-any-address=

May specify a single:

IP address	192.0.2.1
CIDR block	192.0.2.0/24
wildcard pattern	172.16-31.x.1,254
addrs in same subnet	203.0.113.1,3,7,13,19

Specified IP addresses or CIDR blocks

--**s**cidr=	--**d**cidr=	--any-cidr=
--not-**s**cidr=	--not-**d**cidr=	--not-any-cidr=

May specify multiple:

IP addresses	192.0.2.1**,**198.51.100.3
CIDR blocks	192.0.2.0/24**,**198.51.100.0/24
mixture	192.0.2.1**,**192.0.2.8/29

NO wildcard patterns

Sets of arbitrary addresses

--**s**ipset= --**d**ipset= --anyset=

--not-**s**ipset= --not-**d**ipset= --not-anyset=

Specifies the name of a file storing the IP set:

--**s**ipset=internalservers.set

--**d**ipset=RussianBizNtwk.set

--anyset=TorNodes.set

--not-**d**ipset=whitelist.set

Combinations of key fields – Tuples

--tuple-file=TorAuthSockets.tuple --tuple-dir=reverse

TorAuthSockets.tuple file:

sIP	sPort
208.83.223.34	443
82.94.251.203	80
193.23.244.244	80
194.109.206.212	80
86.59.21.38	80
128.31.0.34	9131
171.25.193.9	443
154.35.32.5	80
212.112.245.170	80
76.73.17.194	9030

rwfilter output options

```
--pass-destination=      # file to get records that pass
--fail-destination=      # file to get records that fail
--all-destination=       # file to get all records

--print-statistics       # report recs read/pass/fail
--print-volume-statistics      # report how many
                               # recs/pkts/bytes pass/fail
```

What Is This? — 5

```
rwfilter --sensor=S0 --type=in \
--start=2009/4/21T00 --end=2009/4/21T07 \
--daddress=10.1.0.0/16 --print-volume-stat
```

```
     |  Recs|  Packets|   Bytes|  Files|
Total|  1436|     2615|  158084|      8|
 Pass|  1436|     2615|  158084|       |
 Fail|     0|        0|       0|       |
```

rwfilter exercise

1) Find all traffic captured by sensor S0 going outbound to external HTTPS servers on April 21, 2009. Save these flows in file https0421.rw

2) How many flow records matched the criteria?

rwfilter exercise solution

```
rwfilter --sensor=S0 --type=outweb \
   --start=2009/4/21 --dport=443 \
   --pass=https0421.rw --print-volume-statistics
      |  Recs|Packets|    Bytes|Files|
Total| 43656| 173550| 36174384|   24|
 Pass|   123|   1420|   288083|     |
 Fail| 43533| 172130| 35886301|     |
rwfileinfo https0421.rw --fields=count
https0421.rw:
   count-records        123
```

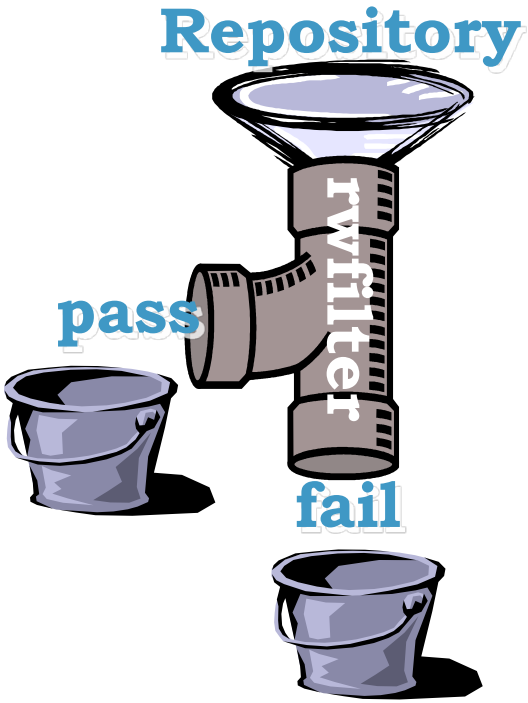
Output Criteria

rwfilter leaves the flows in binary (compact) form.

- **`--pass`**, **`--fail`**: direct the flows to a file or a pipe
- **`--all`**: destination for everything pulled from the repository
- One output is required but more than one can be used (no screen allowed).

Other useful output

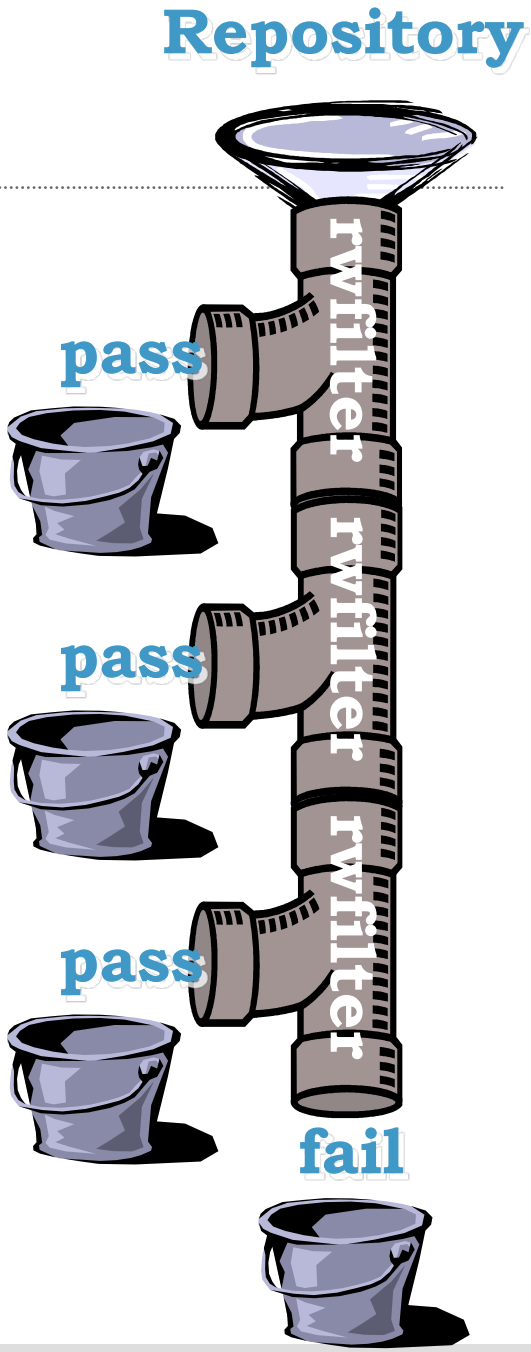
- **`--print-filenames`**, **`--print-missing-files`**
- **`--print-statistics`** or **`--print-volume-statistics`**



Chaining Filters

It is often very efficient to chain rwfilter commands together:

- Use **`--pass`** and **`--fail`** to segregate bins.
- Use **`--all`**, so you only pull from the repository once.



What Is This? — 8

```
rwfilter \
      --start-date=2010/12/08 \
      --type=outweb \
      --bytes=100000- \
      --pass=stdout \
| rwfilter \
      stdin \
      --duration=60- \
      --pass=long-http.rw \
      --fail=short-http.rw
```

Tips with `rwfilter`

Narrow time, type, and sensor as much as possible (fewer records to check).

Include as many partitioning parameters as possible (easy to be vague and get too much data).

Can do multiple queries and merge results

Can do further filtering to narrow results

Iterative exploration

Example Typos

--port= --destport= --sip= or --dip=	No such keywords
--saddress=danset.set	Needs value not filename
--start-date=2006/06/12--end-date	Spaces needed
--start-date = 2006/06/12	No spaces around equals
start-date=2006/06/12	Need dashes
---start-date=2006/06/12	Only two dashes
--start-date=2005/11/04:06:00:00 --end-date=2005/05/21:17:59:59	Only down to hour

SiLK Commandments

1. **Thou shalt use Sets instead of using several rwfilter commands to pull data for multiple IP addresses**
2. **Thou shalt store intermediate data on local disks, not network disks.**
3. **Thou shalt make initial pulls from the repository, store the results in a file, and work on the file from then on. The repository is slower than processing a single file.**
4. **Thou shalt work in binary for as long as possible. ASCII representations are much larger and slower than the binary representations of SiLK data.**
5. **Thou shalt filter no more than a week of traffic at a time. The filter runs for excessive length of time otherwise.**
6. **Thou shalt only run a few rwfilter commands at once.**
7. **Thou shalt specify the type of traffic to filter. Defaults work in mysterious ways.**
8. **Thou shalt appropriately label all output.**
9. **Thou shalt check that SiLK does not provide a feature before building your own.**

Basic SiLK Counting Tools: `rwcount`, `rwstats`, `rwuniq`

"Count [volume] by [key field] and print [summary]"

- basic bandwidth study:
  - "Count bytes by hour and print the results."
- top 10 talkers list:
  - "Count bytes by source IP and print the 10 highest IPs."
- user profile:
  - "Count records by dIP-dPort pair and print all the pairs."
- potential scanners:
  - "Count unique dIPs by sIP and print the sources that contacted more than 100 destinations."

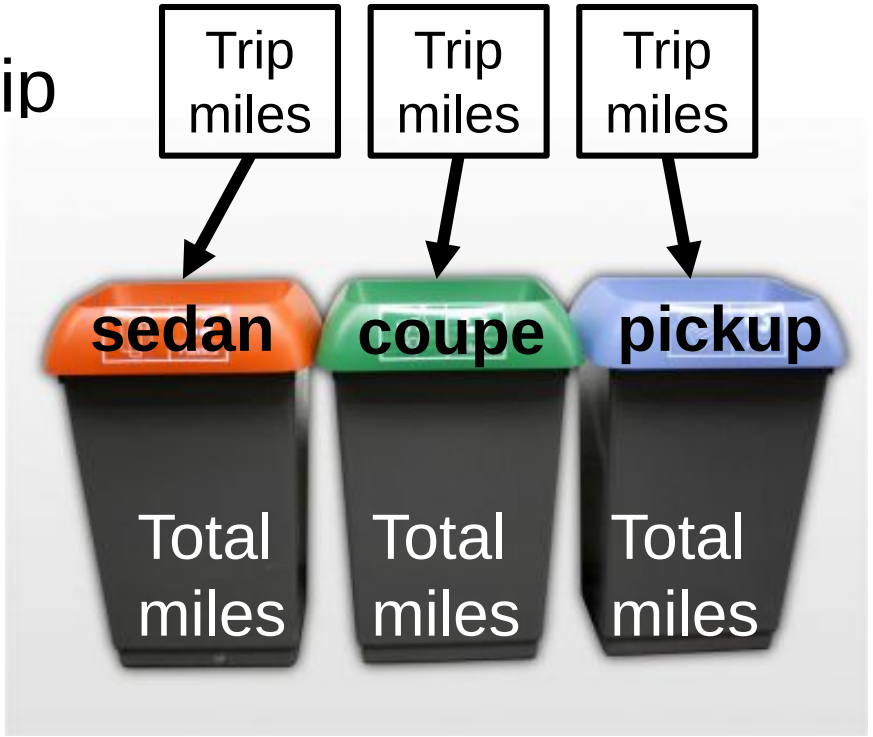
Bins

For motor vehicle trips we could bin by:

- Vehicle style – sedan, coupe, SUV, pickup, van
- Highway or city trip
- Personal or business trip

We could measure the trips and aggregate in bins:

- total miles
- fuel consumption
- oil consumption
- pollutant emission



http://www.prlog.org/10991533-great-value-good-looking-colour-coded-recycling-bins-exclusive-to-imrubbishcouk.html

Bins

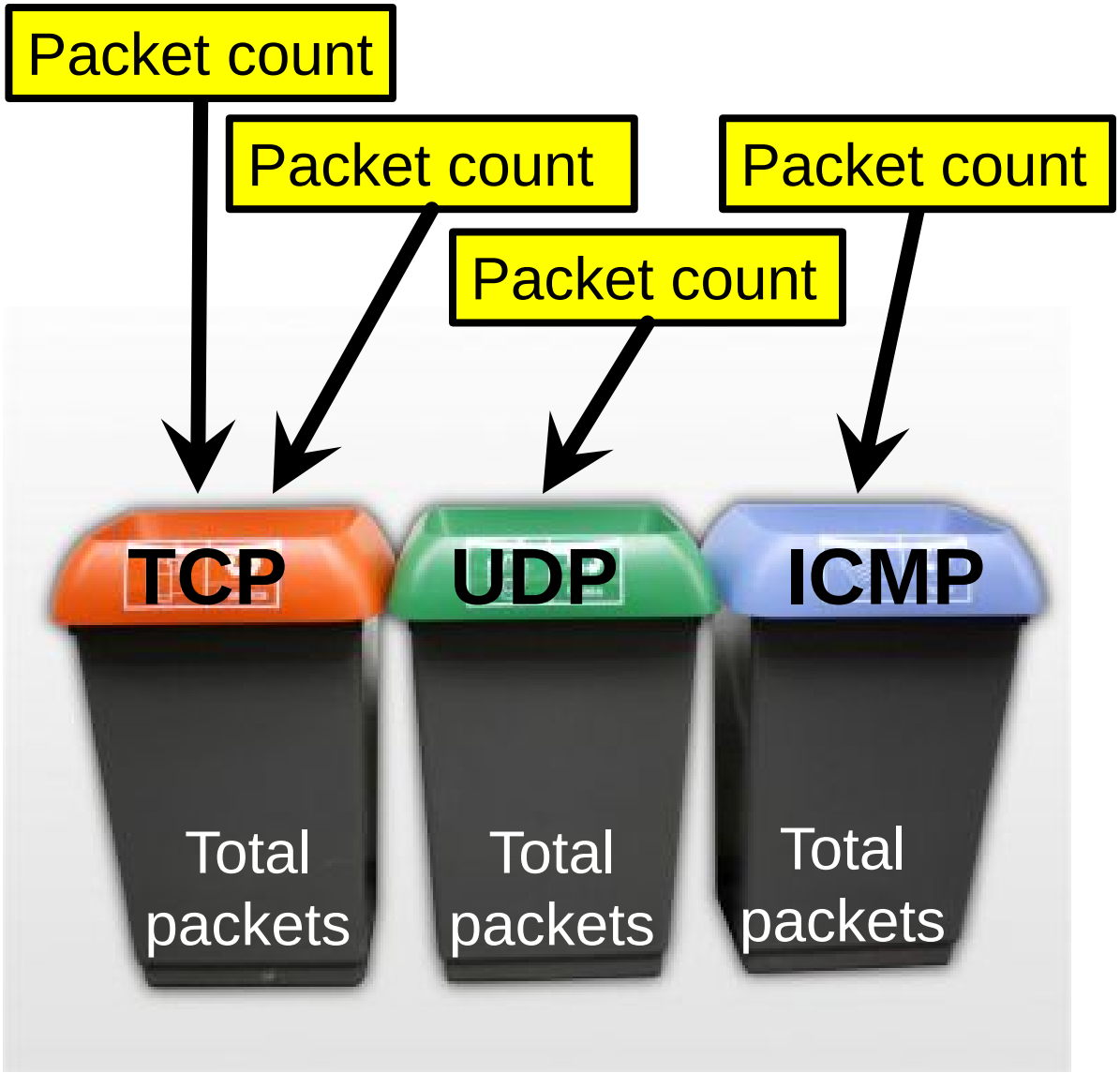
For flows we could bin by:

- address or address block
- port
- protocol
- time period

We could measure the flows and aggregate in bins:

- count of flow records, packets, bytes
- count of distinct values of other fields, e.g., addr
- earliest sTime, latest eTime

Bins



Value
from flow record
e.g., packets

Bin key field
e.g., protocol

Aggregate Value

Basic SiLK Counting Tools: `rwcount, rwstats, rwuniq`

`rwcount`: count volume across time periods

`rwstats`: count volume across IP, port, or protocol and create descriptive statistics

`rwuniq`: count volume across any combination of SiLK fields

“Key field” = SiLK fields defining bins

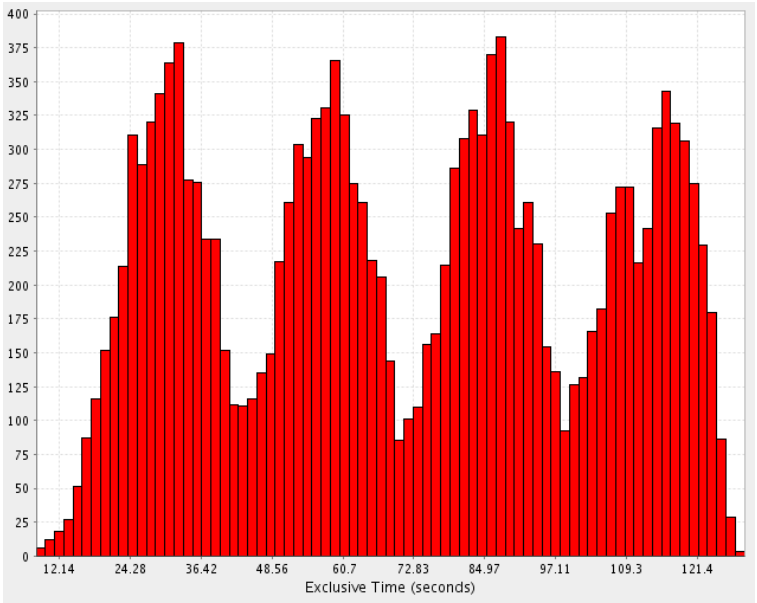
“Volume” = {Records, Bytes, Packets} and a few others

- measure
- aggregate value

Each tool reads raw binary flow records as input.

rwcount

- count records, bytes, and packets by time and display results
- fast, easy way of summarizing volumes as a time series
- great for simple bandwidth studies
- easy to take output and make a graph with graphing S/W



http://www.cs.uoregon.edu/research/tau/docs/paraprof/ch05s02.html

Time Bins

When binning by time, you must specify the period of time for each bin. This is called the **bin-size**.
It's the size of the bin's opening, not the volume of the container.

← bin-size →

rwcount

The bin key is always time. You choose the period.

The aggregate measures are chosen for you. They are flows/records, bytes, packets.

```
rwfilter --sensor=S0 --start=2009/4/21 \
    --type=in --proto=1 --pass=stdout  \
 | rwcount --bin-size=3600
                Date|Records|   Bytes|Packets|
...
2009/04/21T13:00:00|  10.00| 2460.00|  41.00|
2009/04/21T14:00:00|  29.00| 8036.00| 107.00|
2009/04/21T15:00:00|  22.00| 2214.00|  47.00|
2009/04/21T16:00:00|  10.00| 1586.00|  23.00|
...
```

What Is This? — 9

```
rwcount MSSP.rw --bin-size=3600
                Date|    Records|       Bytes|    Packets|
2010/12/08T00:00:00| 1351571.66| 73807086.40| 1606313.61|
2010/12/08T01:00:00| 1002012.43| 54451440.59| 1185143.62|
2010/12/08T02:00:00| 1402404.61| 77691865.26| 1675282.27|
2010/12/08T03:00:00| 1259973.65| 68575249.90| 1491393.08|
2010/12/08T04:00:00|  939313.56| 51410968.24| 1118584.81|
2010/12/08T05:00:00|  459564.75| 80862273.32| 1742058.62|
2010/12/08T06:00:00| 1280651.23| 69881126.41| 1519435.24|
...
```

rwcount Demo

The shell can help with the arithmetic: $((24*60*60))

You also can find common periods in the Quick Reference Guide.

Time series for all outgoing traffic on S0:

```
rwfilter --sensor=S0 --type=out,outweb \
    --start=2009/04/21 --end=2009/04/23 \
    --proto=0- --pass=stdout \
| rwcount --bin-size=$((24*60*60))
```

rwcount Exercise

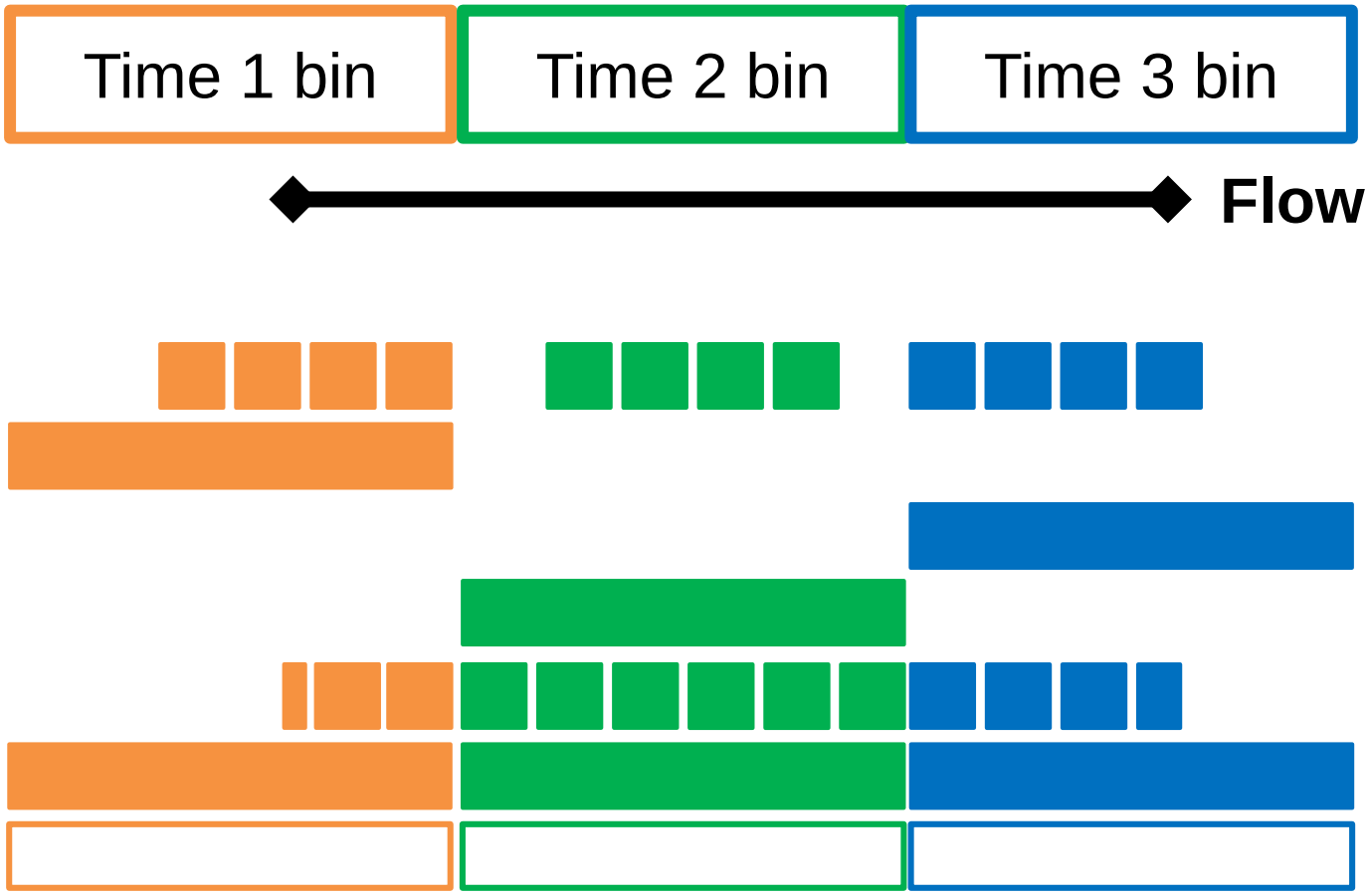
Produce a time-series with 30-minute intervals, analyzing incoming ICMP traffic collected at sensor S0 on April 21, 2009.

rwcount Exercise solution

```
rwfilter --sensor=S0 --type=in,inicmp \
    --start=2009/04/21 --proto=1 \
    --pass=stdout \
| rwcount --bin-size=1800
                Date|Records|   Bytes|Packets|
...
2009/04/21T13:00:00|   5.00|  960.00|  16.00|
2009/04/21T13:30:00|   5.00| 1500.00|  25.00|
2009/04/21T14:00:00|  22.00| 3900.00|  65.00|
2009/04/21T14:30:00|   7.00| 4136.00|  42.00|
2009/04/21T15:00:00|   6.00|  364.00|  13.00|
2009/04/21T15:30:00|  16.00| 1850.00|  34.00|
2009/04/21T16:00:00|   8.00|  934.00|  19.00|
...
```

rwcount --load-scheme

How are records, packets, and bytes allocated to flows that span time bins?



rwcount --load-scheme

How do I choose a loading scheme? (hint: use default)

0 – Average load/bin (smooth peaks/valleys among bins)

1 – Flow onset / periodic behavior emphasis

2 – Emphasize flow termination

3 – Emphasize payload transfer above setup/termination

4 – Average load/time (smooth peaks/valleys over time)

5 – Worst case service loading

6 – Best case service loading

Most commonly used schemes are: **4**, 0, 1

Calling `rwstats`

`rwstats --overall-stats`

- Descriptive statistics on byte and packet counts by record
- See “man rwstats” for details.

`rwstats --fields=KEY --value=VOLUME --count=N or --threshold=N or --percentage=N [--top or --bottom]`

- Choose one or two key fields.
- Count one of records, bytes, or packets.
- Great for Top-N lists and count thresholds
- (standard output formatting options – see “man rwstats”)

What Is This? – 10

```
rwfilter outtraffic.rw \
    --stime=2010/12/08T18:00:00-2010/12/08T18:59:59 \
    --pass=stdout \
 | rwstats --fields=sip --values=bytes --count=10
INPUT: 1085277 Records for 1104 Bins and 4224086177 Total Bytes
OUTPUT: Top 10 Bins by Bytes
                sIP|      Bytes|    %Bytes|   cumul_%|
         71.55.40.62| 1754767148| 41.541935| 41.541935|
        71.55.40.169| 1192063164| 28.220617| 69.762552|
        71.55.40.179|  331310772|  7.843372| 77.605923|
        71.55.40.204|  170966278|  4.047415| 81.653338|
      177.249.19.217|  122975880|  2.911301| 84.564639|
         71.55.40.72|  110726717|  2.621318| 87.185957|
        71.55.40.200|  101593627|  2.405103| 89.591060|
      177.71.129.255|   40166574|  0.950894| 90.541954|
         71.55.40.91|   35316554|  0.836076| 91.378030|
     149.249.114.204|   26634602|  0.630541| 92.008571|
```

rwstats Exercise 1

What are the top 10 incoming protocols on April 22, 2009, collected on sensor S0?

rwstats Exercise 1 solution

```
rwfilter --sensor=S0 --type=in,inweb \
  --start=2009/04/22 --prot=0- --pass=stdout \
| rwstats --fields=protocol --value=rec --count=10
INPUT: 337595 Records for 4 Bins and
337595 Total Records
OUTPUT: Top 10 Bins by Records
pro|   Records|  %Records|   cumul_%|
  6|    336037| 99.538500| 99.538500|
 17|      1467|  0.434544| 99.973045|
  1|        88|  0.026067| 99.999111|
132|         3|  0.000889|100.000000|
```

rwstats Exercise 2

Top 10 inside hosts according to how many outside hosts they communicate with.

Use **--value=distinct:dip**

rwstats Exercise 2 solution

```
rwfilter --sensor=S0 --type=out,outweb --proto=0- \
    --start-d=2009/4/22 --pass=stdout \
| rwstats --fields=sip --value=distinct:dip --count=10
INPUT: 313028 Records for 7 Bins
OUTPUT: Top 10 Bins by dIP-Distinct
         sIP|dIP-Distin|%dIP-Disti|cumul_%|
10.1.60.187|         50|          ?|      ?|
  10.1.60.5|         26|          ?|      ?|
 10.1.60.25|         17|          ?|      ?|
 10.1.60.73|         14|          ?|      ?|
10.1.60.191|         11|          ?|      ?|
10.1.60.251|          9|          ?|      ?|
10.1.60.132|          3|          ?|      ?|
```

--no-percents will clean up the question marks.

rwuniq

Unlike rwstats, rwuniq will display all the bins, not just the top or bottom N bins.

Output is normally unsorted. --sort-output causes sorting by the key (bin), unlike rwstats which sorts by aggregate value.

Calling `rwuniq`

`rwuniq --fields=KEY --value=VOLUME`

- Choose one or several key fields.
- Aggregate volume count: records, bytes, or packets.
- (standard output formatting options – see “man rwuniq”)

Apply thresholds to bins before outputting:

- **`--bytes`**, **`--packets`**, **`--flows`**, **`--sip-distinct`**, **`--dip-distinct`**
- Specify minimum aggregate value or a range

`--sort-output` by key (**`rwstats`** sorts by value)

What Is This? – 11

```
rwfilter outtraffic.rw \
    --stime=2010/12/08:18:00:00-2010/12/08:18:59:59 \
    --saddress=71.55.40.62 --pass=stdout \
 | rwuniq --fields=dip,sport --all-counts --sort-output
```

dIP	sPort	Bytes	Packets	Records	sTime-Earliest	eTime-Latest
12.113.41.190	80	12782	20	4	2010/12/08T18:42:51	2010/12/08T18:58:49
30.182.228.143	80	203907933	143611	2	2010/12/08T18:53:59	2010/12/08T19:01:47
37.153.24.229	80	205628625	144829	2	2010/12/08T18:29:11	2010/12/08T18:42:51
82.180.203.87	80	213013145	150896	92	2010/12/08T18:06:36	2010/12/08T18:32:33
82.180.203.197	80	800	8	2	2010/12/08T18:43:30	2010/12/08T18:43:30
88.124.166.233	80	223930369	158276	97	2010/12/08T18:08:55	2010/12/08T18:32:25
88.124.166.233	443	509285	732	43	2010/12/08T18:06:57	2010/12/08T18:51:11
94.239.226.247	80	124833037	96047	3	2010/12/08T18:25:22	2010/12/08T19:21:34
109.95.61.80	80	8467397	6325	90	2010/12/08T18:08:59	2010/12/08T18:10:09
139.65.186.4	80	204123360	143794	3	2010/12/08T18:19:48	2010/12/08T18:26:36
139.177.10.136	80	407978375	287354	6	2010/12/08T18:20:03	2010/12/08T19:01:30
198.237.16.172	80	159066748	112025	1	2010/12/08T18:18:43	2010/12/08T18:46:55
219.149.72.154	1024	44	1	1	2010/12/08T18:50:40	2010/12/08T18:50:40
249.216.88.172	80	88	2	2	2010/12/08T18:44:42	2010/12/08T18:44:47
250.211.100.88	80	3295160	2492	42	2010/12/08T18:47:50	2010/12/08T18:58:53

What Is This? – 12

```
rwuniq outtraffic.rw --fields=dip \
 --values=sip-distinct,records,bytes --sip-distinct=400- \
 --sort-output
              dIP|sIP-Distin|     Bytes| Records|
   13.220.28.183|         512|     20480|     512|
    171.128.2.27|         448|  19069280|  476732|
   171.128.2.179|         448| 139501200| 3487530|
  171.128.212.14|         448| 139467440| 3486686|
 171.128.212.124|         448| 127664480| 3191612|
 171.128.212.127|         448|  66611560| 1665289|
 171.128.212.188|         448| 139467680| 3486692|
 171.128.212.228|         448| 139393160| 3484829|
 245.225.153.120|         763|     30520|     763|
 245.238.193.102|        1339|    179480|    4487|
```

rwuniq vs. rwstats

rwuniq	both	rwstats in top/bottom mode
all bins except per thresholds	Bin by key	**--top** or **--bottom** bins
	Default aggregate value is flows (records)	
--sort-output by key otherwise unsorted		Sorted by primary aggregate value
Thresholds or ranges: --bytes, --packets, --flows, --sip-distinct, --dip-distinct	Choose which bins have aggregate values significant enough to output.	**--count**, **--threshold**, **--percentage**
--all-counts (bytes, pkts, flows, earliest sTime, and latest eTime)	Show volume aggregate value[s]	**--no-percents** (good when primary aggregate isn't Bytes, Packets, or Records)
	--bin-time to adjust sTime and eTime	
	--presorted-input (omit when value includes Distinct fields, even if input is sorted)	
--values=sTime-Earliest, eTime-Latest	**--values**=Records, Packets, Bytes, sIP-Distinct, dIP-Distinct, Distinct:KEY-FIELD (KEY-FIELD can't also be key field in --fields)	

Blacklists, Whitelists, Books of Lists...

Too many addresses for the command line?

- spam block list
- malicious websites
- arbitrary list of any type of addresses

Create an IP set!

- individual IP address in dotted decimal or integer
- CIDR blocks, 192.168.0.0/16
- wildcards, 10.4,6.x.2-254

Use it directly within your filter commands.

- **`--sipset`**, **`--dipset`**, **`--anyset`**

Set Tools

`rwsetbuild`: Create sets from text.

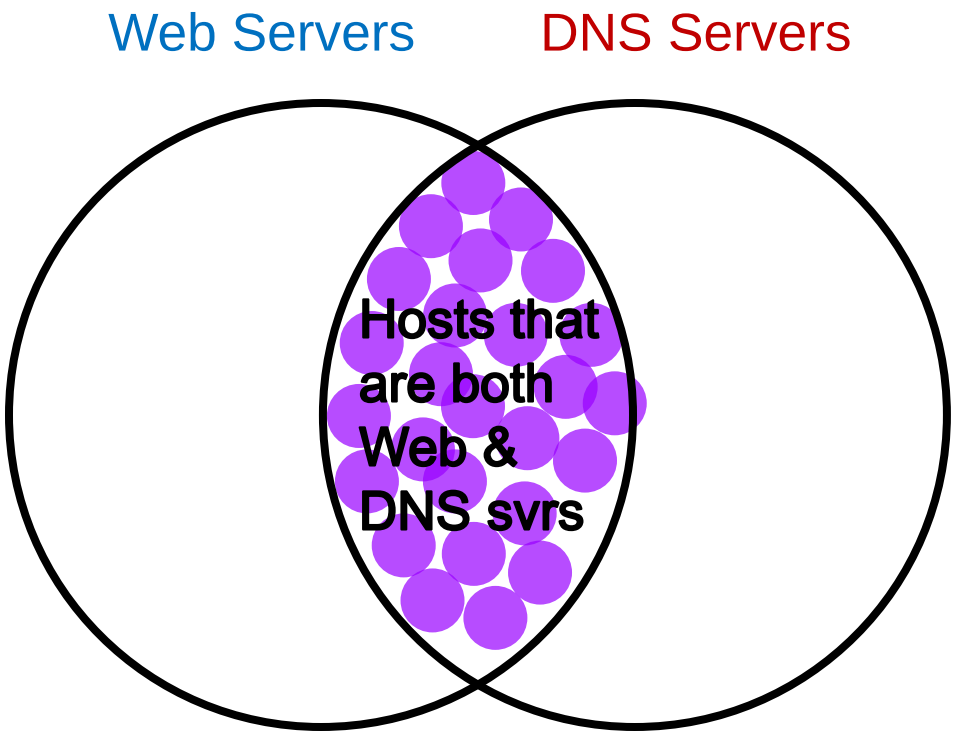
`rwset`: Create sets from binary flow records.

`rwsetcat`: Display an IP set as text.

`rwsetmember`: Test if an address is in given IP sets.

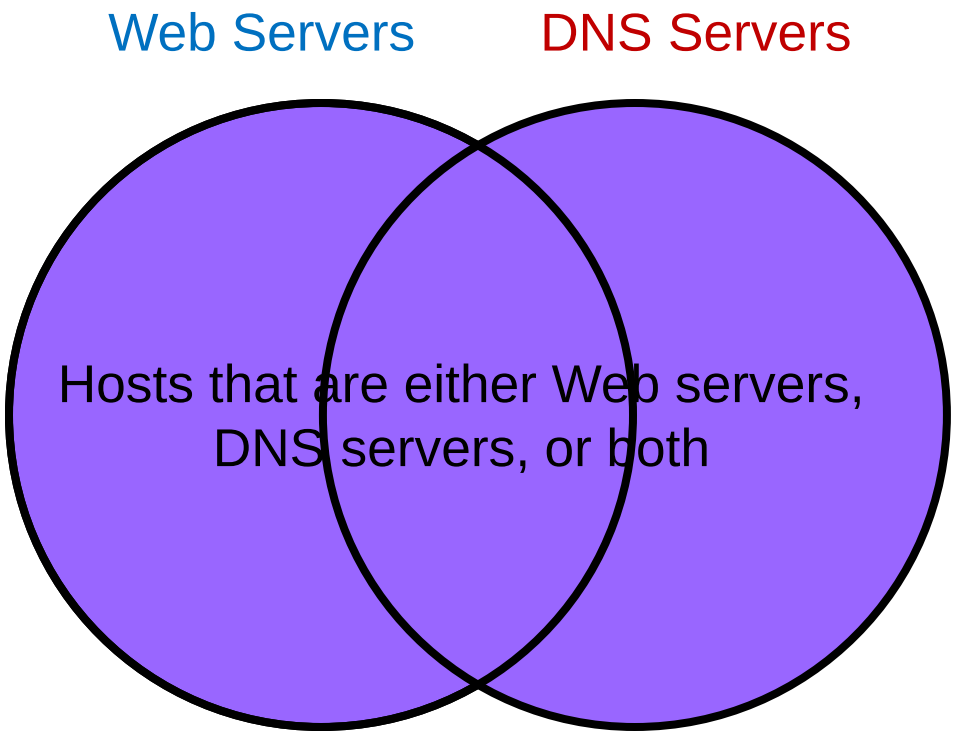
`rwsettool`: Perform set algebra (intersection, union, set difference) on multiple IP sets.

Set Intersection



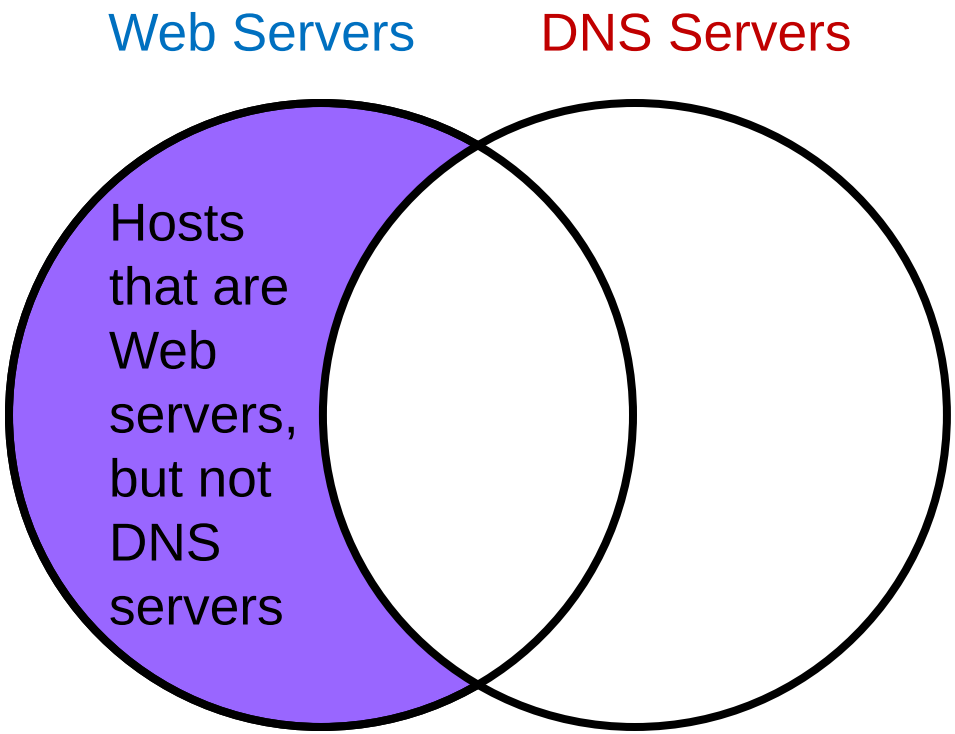
rwsettool --intersect web.set dns.set --output web_and_dns.set

Set Union



rwsettool --union web.set dns.set --output web_or_dns.set

Set Difference




```
rwsettool --difference  web.set  dns.set  --output  web_not_dns.set
```

What Is This? — 6

```
more MSSP.txt
171.128.2.0/24
171.128.212.0/24

rwsetbuild MSSP.txt MSSP.set
rwfilter --start=2010/12/8 --anyset=MSSP.set \
  --pass=MSSP.rw --print-vol
      |       Recs|     Packets|         Bytes|Files|
Total|   30767188|    81382782|    35478407950|   48|
 Pass|   26678669|    31743084|     1464964676|     |
 Fail|    4088519|    49639698|    34013443274|     |

rwset --sip-file=MSSPsource.set MSSP.rw
rwsettool --intersect MSSP.set MSSPsource.set \
   --output=activeMSSP.set
rwsetcat --count-ips activeMSSP.set
22
```

What Is This? — 7

```
rwfilter --sensor=S0 --type=out \
 --start=2009/4/21 --proto=0- \
 --pass=stdout \
 | rwset --dip-file=outIPs.set
rwsetcat outIPs.set --network-structure=16
      10.1.0.0/16| 8748
      10.2.0.0/16| 27
    140.13.0.0/16| 1
```

Set Exercise 1

Make a set-file of addresses of all actual inside hosts.

Should we examine incoming or outgoing traffic?

Make a set-file of all outside addresses.

Can you make both sets with one command?

Set Exercise 1 solution

```
rwfilter --sensor=S0 --type=out,outweb \
    --start-d=2009/4/21 --end=2009/4/23 \
    --proto=0- --pass=stdout \
| rwset --sip-file=insidehosts.set \
         --dip-file=outsidehosts.set
```

Set Exercise 2

Examine the two set-files.

Set Exercise 2 solution

```
ls –l insidehosts.set
rwfileinfo insidehosts.set
rwsetcat insidehosts.set

ls –l outsidehosts.set
rwsetcat outsidehosts.set | less
```

Set Exercise 3

Which /24 networks are on the inside?

Which /24 networks are on the outside?

Set Exercise 3 solution

```
rwsetcat --network-struc=24 insidehosts.set
rwsetcat --network-struc=24 outsidehosts.set
```

Advanced Partitioning Options

- TCP Flags
- Count of packets and bytes
- Time
- Extending rwfilter's partitioning options with plugins

TCP Flags

S – Syn (synchronize)

U – Urg (urgent)

R – Rst (reset)

F – Fin (finish)

P – Psh (push)

A – Ack (acknowledge)

C – CWR (congestion window reduced)

E – ECE (explicit congestion notification echo)

TCP Flags

--flags-initial= # TCP flags in 1st pkt of flow

--flags-session= # flags in remaining packets

--flags-all= # flags in all pkts of flow

=flagsOn/flagsExamined

flagsOn: TCP flags that must be On to pass.

flagsExamined: flags under consideration for passing.

Any flags in flagsOn must also be in flagsExamined.

Flags in flagsExamined, but not in flagsOn, must be off to pass.

TCP Flags

--flags-initial=S/SA # flow from client to server

--flags-initial-SA/SA # flow from server to client

--flags-init=S/SA --flags-session=F/F #full C->S flow

--flags-init=SA/SA --flags-session=F/F #full S->C flow

--flags-all=S/SFR # incomplete flow

--flags-all=/FR # unfinished flow fragment

Count of Packets and Bytes

--packets= # packets in the flow

--bytes= # bytes in the packets in flow

--bytes-per-packet= # average

--packets=3-

--bytes=40-570

--bytes-per-packet=40.0-75.125

Partitioning by Time

--stime=*earliertime-latertime*

--etime=*earliertime-latertime*

--active-time=*earliertime-latertime*

--duration=*lowseconds-highseconds*

stime and etime are usually **not** used together.

Each time has millisecond resolution.

--stime=2009/4/21T13:00-2009/4/21T13:29 # ½ hr

--etime=2009/4/21T13:00:00-2009/4/21T13:00:09 # 10 sec

--stime=2009/4/21T13:00-2009/4/21T13:00:48.725 # 48.725s

Extending Partitioning with Plugins

rwfilter‘s partitioning capabilities can be extended with plugins written in Python or C.

```
--python-expr=    # simple python expression
--python-file=    # complex python pgm in a file
--plugin=         # compiled C program in a file

--python-expr='rec.sport == rec.dport'
--python-file=clientserver_filt.py
--plugin=app-mismatch.so
```

I Only Believe What I See

You'll be tempted to work with text-based records.

- It's easy to see the results and post-process with other tools (e.g., Perl, awk, sed, sort).
- It takes a lot of space, and it's ***much, much*** slower.

Guiding principle: Keep flows in binary format as long as possible.

What Is This? — 13

```
rwfilter --type=out  \
   --start=2010/12/08 \
   --aport=22 --pass=ssh.rw

rwfilter --dport=22 ssh.rw \
   --pass=stdout | rwcut

rwfilter --sport=22 ssh.rw \
   --pass=stdout | rwcut
```

Outline — 4

Introduction: SiLK

Network flow

Basic SiLK tools

Advanced SiLK tools

Summary

PySiLK—Using SiLK with Python

- PySiLK—an extension to Python
- Allows Python to manipulate SiLK's data files
- Uses the "silk" python module, from SEI CERT.

PySiLK components

PySiLK

- Read, manipulate, and write SiLK Flow records, IPsets, Bags, and Prefix Maps (pmaps) from within Python

SilkPython (--python-file=)

- Create plug-ins for rwfilter or other SiLK utilities.
  - Create partitioning switches for rwfilter
  - Create new flow-record fields for other utilities

--python-expr=

- Create a simple partitioning test without creating a new switch

Stand-alone PySiLK example

```
#! /usr/bin/env python
import silk
myfile = silk.silkfile_open("MyFlows.rw", silk.READ)
for rec in myfile:
        if rec.sport < 2500 and rec.sport == rec.dport:
        print rec.sport, rec.stime, rec.sip, rec.dip
myfile.close()
```

PySiLK exercise

Write a Python program which reports the source IP address associated with the lowest source port used by any flow record in the file MyFlows.rw.

PySiLK exercise solution

```
#! /usr/bin/env python
import silk
lowsport = 65536  # could use 99999
myfile = silk.silkfile_open("MyFlows.rw", silk.READ)
for rec in myfile:
        if rec.sport < lowsport:
                lowsport = rec.sport
                lowsip = rec.sip
myfile.close()
print rec.sport, rec.sip
```

--python-expr example

```
rwfilter sample.rw \
        --protocol=6 \
        --python-expr='rec.sport == rec.dport' \
        --pass=equalTCPports.rw
```

SilkPython example (1)

```
import silk
def lowerport(rec):
        if rec.sport < rec.dport:
                return rec.sport
        else:
                return rec.dport
register_int_field("lport", lowerport,  0, 65535)
```

SilkPython example (2)

```
rwstats --python-file=lowport.py --fields=lport \
        --value=records --count=10 flows.rw
```

SilkPython exercise

Write a plug-in for rwcut, rwstats, etc. The plug-in should define a new flow-record field which contains the IP address of the host using the lower port number in the flow. You'll need the following SilkPython function:

register_ip_field(field_name, ip_function)

SilkPython exercise solution

```
import silk
def lowerport_ip(rec):
        if rec.sport < rec.dport:
                return rec.sip
        else:
                return rec.dip
register_ip_field("lip", lowerport_ip)
```

Alternatives to PySiLK

- SiLK tools
  - Not as flexible criteria as Python.
  - Could use tuple files
    - Must be maintained
    - Aren’t self-contained with logic
    - Large tuple files run slower than Python.
- Text processing with Perl, C, or Java
  - Create text with rwcut delimited without titles
  - Convert ports back to integers
  - Dealing with dates, times, or addresses difficult

Modified example of PySilk

- Summarize the selection as a count by port
- Just keep a Python dictionary
  - Key = port number
  - Value = count

PySiLK advantages

- Speeds both programming and processing
  - Keeps data in binary, unlike Perl & C
    - No parsing text
  - Built-in conversions of objects to strings
  - Full power of Python
- Good for:
  - Stateful filters and output options
  - Integrate SiLK with other data types
  - Complex or branching filter rules
  - Custom key fields and aggregators for rwcut, rwsort

Outline — 5

Introduction: SiLK

Network flow

Basic SiLK tools

Advanced SiLK tools

Summary

Furthering Your SiLK Analysis Skills (1)

Each tool has a **`--help`** option.

SiLK Reference Guide

SiLK Analysts' Handbook

- Both available at the SiLK tools website http://tools.netsa.cert.org

Email support

- silk-help@cert.org

Furthering Your SiLK Analysis Skills (2)

Tool tips

- SiLK Tooltips link on http://tools.netsa.cert.org

Flow analysis research and advanced techniques

- http://www.cert.org/flocon
- http://www.cert.org/netsa

Questions?

Contact Information

Ron Bandes — **rbandes@cert.org**
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA